KARPATHY DÖNGÜSÜ

Yazılım 2.0 ve Kendini İyileştiren Yapay Zekâ Sistemleri

Necmi Gürsakal & ChatGPT

Bursa-2026

KARPATHY DÖNGÜSÜ

Yazılım 2.0 ve Kendini İyileştiren Yapay Zekâ Sistemleri

Bursa-2026

İÇİNDEKİLER

GİRİŞ

1. TARİHSEL TEMELLER: DÜŞÜNMEYİ İNŞA ETMEK
2. SİSTEM FİKRİNİN DOĞUŞU
3. BİLİŞSEL DEVRİM
4. SEMBOLİK ZEKÂ VE KLASİK YZ
5. KRİZ: SEMBOLİK ZİHNİN SINIRLARI
6. MODERN KIRILMA: DERİN ÖĞRENME
7. KARPATHY VE METİN ÜRETİM SİSTEMİ
8. ÖZYİNELEMELİ KENDİNİ İYİLEŞTİRME
9. YAZILIM 2.0: KARPATHY DÖNGÜSÜ VE YAPAY ZEKÂNIN GELECEĞİ
10. UYGULAMALAR

SONUÇ

GİRİŞ

Bu kitabın çıkış noktası, Andrej Karpathy'nin modern yapay zekâ sistemlerini yalnızca bir "model" olarak değil, aynı zamanda bir "metin üretim sistemi" olarak ele alma biçimidir. Bu bakış, ilk anda teknik bir öneri gibi görünse de aslında çok daha derin bir zihinsel kırılmaya işaret eder: düşünmenin kendisini yazılım benzeri, katmanlı ve çalıştırılabilir bir sistem olarak görme fikrine. Karpathy'nin pratikte önerdiği şey, yalnızca daha iyi istem, talimat veya prompt yazmak ya da daha iyi modeller kullanmak değildir; düşünme sürecini baştan sona yeniden organize etmektir. Yapay zekâ modelinden net olarak ne istediğimiz, belirtme (spesifikasyon), doğrulama ve çalışma zamanı gibi katmanlara ayrılmış bu yaklaşım, insan zihninin kendi üretim biçimini de dolaylı olarak yeniden tanımlar.

Bunu yaptığımızda elde edilen şey aslında "daha iyi cevaplar" değil, çok daha temel bir şeydir: düşünmeyi kontrol edilebilir bir üretim sürecine dönüştürmek.

Bunu biraz açalım.

Karpathy'nin önerdiği çerçeve (belirtme–doğrulama–çalışma zamanı), ilk bakışta yazılım mühendisliğine ait gibi görünür. Ama bunu düşünmeye uyguladığımızda ortaya çıkan sonuç şudur: düşünme artık tek bir zihinsel akış değil, yönetilebilir katmanlara ayrılmış bir sistem davranışı haline gelir.

Bu ne kazandırır?

İlk olarak, belirsizlik dramatik biçimde azalır. Geleneksel düşünme ya da yazma sürecinde en büyük problem şudur: ne istediğimiz net değildir, üretim sırasında şekillenir. Bu da dağınık, tekrarlı ve hataya açık bir süreç yaratır. Belirtme katmanı bunu tersine çevirir: önce düşünce alanını daraltır, ne üretileceğini netleştirir. Böylece zihin "ne yapıyorum ben?" sorusuna daha baştan cevap verir.

İkinci olarak, hata artık sona bırakılmaz, sürece yayılır. Doğrulama katmanı sayesinde üretim sırasında sürekli bir kontrol mekanizması çalışır. Bu, düşünmeyi "üret sonra düzelt" modelinden çıkarıp "üretirken düzelt" modeline taşır. Sonuç, daha az geri dönüş, daha az zihinsel maliyet ve daha tutarlı çıktıdır.

Üçüncü olarak, çalışma zamanı (runtime) katmanı düşünmeyi statik bir plan olmaktan çıkarır. Artık düşünce, kâağıt üzerinde tamamlanmış bir şey değil; çalışırken şekillenen bir süreçtir. Bu da çok kritik bir dönüşümdür çünkü gerçek karmaşıklık, ancak çalışırken ortaya çıkar. Yani

sistemin gerçek davranışı, tasarım anında değil, çalışma anında görünür hale gelir. Bunların birleşimi şu sonucu doğurur:

Düşünme artık “içsel bir akış” değil, geri beslemeli bir üretim sistemi olur.

Bu çerçeve doğrudan yapay zekâ sistemlerinin modern mühendislikte nasıl inşa edildiğini açıklayan bir modele dönüşür. Özellikle Anthropic’in geliştirdiği sistemlerde bu üç katmanlı yapı yalnızca teorik bir yaklaşım değil, doğrudan üretim hattının kendisidir. Claude gibi modeller, yalnızca bir “cevap üreten sistem” olarak değil, aynı zamanda kendi çıktısını değerlendiren ve gerektiğinde yeniden üreten bir döngü içinde çalışacak şekilde tasarlanır. Bu yapı, Karpathy’nin önerdiği belirtme–doğrulama–çalışma zamanı ayrımının, yazılım seviyesinden model mimarisine taşınmış halidir.

Anthropic’in yaklaşımında belirtim katmanı, yalnızca kullanıcı istemleri, talimatları ile sınırlı değildir; aynı zamanda sistem davranışını tanımlayan daha üst seviye “anayasal kurallar” ile genişletilmiştir. Bu kurallar, modelin hangi tür cevapları üretmesi gerektiğini, hangi tür davranışlardan kaçınması gerektiğini ve hangi sınırlar içinde hareket edeceğini belirler. Böylece modelin üretim alanı baştan tanımlanmış bir düşünce uzayına oturtulur. Bu, Karpathy’nin “ne istiyoruz?” sorusunu yazılım düzeyinden çıkarıp sistem düzeyine taşımanın pratik karşılığıdır.

Doğrulama ve çalışma zamanı katmanları ise Anthropic sistemlerinde daha dinamik bir yapıya sahiptir. Model yalnızca bir çıktı üretmez; bu çıktı, başka bir değerlendirme mekanizması tarafından incelenir, gerekirse yeniden yazılır veya reddedilir. Bu süreç, insan geri bildirimi (RLHF) ve otomatik denetleyici modellerle desteklenir. Böylece sistem, sadece üretim yapan bir yapı olmaktan çıkar, kendi üretimini sürekli kontrol eden ve geri besleme ile şekillendiren bir döngüye dönüşür. Bu noktada Karpathy Döngüsü, Anthropic’in mühendislik pratiğinde yalnızca bir tasarım fikri değil, doğrudan operasyonel bir gerçeklik haline gelmiş olur.

Buradan elde edilen en önemli kazanım şudur: insan zihni kendi kendini daha açık modellemeye başlar. Yani kişi sadece yazı yazmaz, aynı zamanda nasıl düşündüğünü de dışsallaştırır. Bu da metni, notu veya problemi bir “sonuç” olmaktan çıkarıp bir “sistem taslağı” haline getirir.

En derin etkisi ise şudur:

Bu yaklaşım, insan düşünmesini doğrudan yapay zekâ sistemleriyle hizalar. Çünkü model artık sadece bir araç değil, düşünme sürecinin bir bileşeni haline gelir. İnsan ve model birlikte çalıştığında ortaya çıkan şey, tek bir zihnin üretimi değil, insan–makine ortak bilişsel sistemi olur. Kısacası:

Bu çerçeveyle elde edilen şey daha iyi istemler (talimatlar-promptlar) değil, daha temiz cevaplar değil, sadece daha hızlı üretim de değildir. Elde edilen şey: Düşünmenin bir süreç olmaktan çıkıp, katmanlara ayrılmış bir üretim mimarisi haline gelmesidir.

Bu çerçevenin gerçek hayattaki karşılığı, “daha iyi yazı yazmak” gibi dar bir fayda değil; karmaşık işleri daha az zihinsel dağılmayla, daha yüksek kontrolle ve daha düşük hata oranıyla yürütmek olur. Bunu somutlaştırınca üç ana kazanım ortaya çıkar:

İlk olarak, işlerin çözülme biçimi değişir. Normalde bir problemle karşılaştığında insanlar genellikle “düşünmeye başlar ve üretim sürecine girer.” Ama bu süreç çoğu zaman bulanıktır: ne istenildiği net değildir, ilerlerken şekillenir, sonra geri dönülür. Karpathy’nin çerçevesi bunu tersine çevirir. Önce “ne istiyorum?” netleşir (belirtme), sonra üretim yapılır, sonra kontrol edilir. Bu, gerçek hayatta proje yönetiminden yazılım geliştirmeye, akademik yazımdan stratejik kararlara kadar her yerde belirsizliği azaltan bir yapı üretir.

İkinci olarak, insan–YZ iş birliği radikal biçimde verimli hale gelir. Normal kullanımda insanlar modele genellikle “bir şey üret” der ve sonucu pasif şekilde değerlendirir. Bu çerçevede ise model bir “üretici” değil, sürecin bir ortağı haline gelir. Belirtme net olduğunda model çok daha hedefli çalışır, doğrulama katmanı sayesinde hatalar erken yakalanır, çalışma zamanı içinde ise alternatifler hızlıca test edilir. Bu, özellikle rapor yazma, analiz üretme, kod geliştirme, strateji oluşturma gibi alanlarda ciddi zaman kazancı yaratır.

Üçüncü ve daha derin etki ise bilişsel yükün yeniden dağıtılmasıdır. Geleneksel çalışmada insan aynı anda hem ne istediğini düşünür hem üretir hem kontrol eder. Bu zihinsel çakışma verimliliği düşürür. Bu çerçevede ise bu roller ayrışır: biri tanımlar, biri üretir, biri kontrol eder. Bu ayrışma, zihni “aynı anda her şeyi yapmaya çalışan bir sistem” olmaktan çıkarıp daha modüler bir yapıya dönüştürür.

Bunu gerçek hayata indirgediğimizde ortaya çıkan şey şudur:

- Daha net proje tanımları
- Daha az geri dönüş ve revizyon
- Daha hızlı iterasyon döngüleri
- Daha tutarlı çıktı kalitesi
- Ve en önemlisi: daha az zihinsel dağılma

Ama belki de en kritik sonuç şudur: insan artık işi “yapan” değil, işi “tasarlayan ve idare eden” konuma geçer.

Bu yaklaşım özellikle karmaşık işlerde fark yaratır: yazılım geliştirme, araştırma, raporlama, ürün tasarımı, hatta stratejik karar alma süreçleri… Çünkü bu alanların ortak problemi şudur: problem genelde çözülmezlikten değil, yanlış tanımlanmaktan kaynaklanır. Belirtme katmanı bu noktayı doğrudan hedef alır. Özetle gerçek hayattaki etkisi şudur:

Bu çerçeve, insanın çalışma biçimini “akış içinde düşünme”den “yapı kurarak düşünme”ye taşır. Ve bu küçük gibi görünen fark, pratikte çok büyük bir verimlilik ve kontrol artışı üretir.

İnsan düşüncesi uzun süre boyunca bir “iç deneyim”, bir “bilinç hâli” ya da en iyi ihtimalle felsefi bir açıklama nesnesi olarak ele alındı. Düşünmek, insan olmanın doğal ve açıklanamaz bir parçası gibi görüldü; nasıl çalıştığı sorusu ya metafizik düzeyde bırakıldı ya da psikolojinin erken dönemlerinde davranış gözlemlerine indirgenerek dolaylı biçimde ele alındı. Ancak 20. yüzyılın ortasından itibaren bu durum köklü bir dönüşüm geçirmeye başladı. Düşünme artık yalnızca “ne olduğu” sorusuyla değil, “nasıl çalıştığı” ve giderek “nasıl tasarlanabileceği” sorusuyla birlikte ele alınmaya başlandı. Bu dönüşüm, basit bir bilimsel ilerleme değil, zihnin kendisine bakışın değişmesiydi. Düşünme, açıklanacak bir fenomen olmaktan çıkıp mühendislik nesnesine dönüşmeye başladı.

Bu kitap, tam olarak bu kırılmanın içinden yazılmaktadır. “Düşünmenin mühendislik olarak yeniden tanımı” ifadesi, yalnızca bir metafor değildir; zihnin artık bir sistem, bir mimari ve giderek bir üretim altyapısı olarak görülmeye başlanmasının doğrudan ifadesidir. “Düşünmenin mühendislik olarak yeniden tanımı” ifadesi aslında tek bir cümlelik bir metafor değil, zihne bakıştaki bütün paradigma değişimini özetleyen bir kırılma noktasıdır. Burada “mühendislik” kelimesi özellikle kritik, çünkü yalnızca teknik bir üretim biçimini değil, aynı zamanda sistematik tasarım, bileşenlere ayırma, sınır tanımlama ve optimize etme gibi zihni doğrudan yeniden kuran bir yaklaşımı ifade eder.

Klasik felsefi gelenekte düşünme, çoğunlukla içerik üzerinden ele alınırdı: “doğru fikir nedir?”, “gerçeklik nasıl bilinir?”, “akıl nasıl hakikate ulaşır?” gibi sorular merkezdeydi. Bu yaklaşımda düşünme, zaten var olan bir içsel yetinin çalışması olarak görülür. Yani zihin bir anlamda “verili”dir; yapılacak şey onu yorumlamak ve açıklamaktır.

Mühendislik yaklaşımı ise bu noktada tamamen farklı bir başlangıç yapar. Buradaki temel soru “zihin nedir?” değil, “zihin nasıl çalışır ve nasıl yeniden kurulabilir?” sorusudur. Bu, çok kritik bir farktır. Çünkü açıklama dili yerini tasarım diline bırakır. Artık amaç bir fenomeni anlamak değil, o fenomenin çalışabilir bir versiyonunu inşa etmektir.

Bu çerçevede düşünme, tek parça bir süreç olmaktan çıkar ve bir sistem olarak ele alınır. Sistem dediğimiz şey ise parçaların, bu parçalar arasındaki ilişkilerin ve bu ilişkilerden doğan davranışın bütünüdür. Yani düşünme artık “bir şey” değil, bir organizasyondur. Algı, dikkat, bellek, temsil ve karar verme gibi bileşenler bu organizasyonun alt modülleri haline gelir.

Karpathy’nin yaklaşımı bu mühendislik bakışını pratik bir düzeye indirir. Örneğin metin üretimi artık “zihinsel bir ifade” değildir; belirtme (spesifikasyon), doğrulama (verification) ve çalışma zamanı (runtime) gibi katmanlara ayrılmış bir üretim sürecidir. Bu ayrım aslında şunu gösterir: karmaşık bir bilişsel süreç, doğru parçalandığında bir iş akışına dönüşebilir. Ve iş akışı haline gelen bir süreç, artık tasarlanabilir, ölçülebilir ve optimize edilebilir hale gelir.

Buradaki en önemli dönüşüm şudur: düşünme artık içsel bir gizem olmaktan çıkar, dışsal olarak gözlemlenebilir ve yeniden kurulabilir bir sistem haline gelir. Bu, zihni bir “kara kutu” olarak görmekten vazgeçip, onu bir “mimari sistem” olarak ele almak demektir. Kara kutu yaklaşımında yalnızca giriş ve çıkışlar vardır; mühendislik yaklaşımında ise aradaki tüm

katmanlar analiz edilebilir. Dolayısıyla “düşünmenin mühendislik olarak yeniden tanımı”, zihni üç seviyede yeniden çerçeveler:

Birinci seviye, bileşen düzeyidir: düşünme hangi modüllerden oluşur? İkinci seviye, etkileşim düzeyidir: bu modüller nasıl birlikte çalışır? Üçüncü seviye, sistem düzeyidir: bu etkileşimlerden nasıl bir davranış ortaya çıkar?

Bu üçlü yapı, modern yapay zekâ sistemlerinin de temelini oluşturur. Aynı zamanda Karpathy’nin “metin üretim sistemi” yaklaşımının da altında yatan zihinsel model budur: düşünme, tek bir akış değil, katmanlı bir üretim mimarisidir.

Bu nedenle bu ifade yalnızca teknik bir yeniden adlandırma değildir. Aslında çok daha radikal bir şeyi söyler: düşünme, artık keşfedilen bir şey değil, tasarlanan bir şeydir.

Felsefe tarihinde düşünme çoğunlukla içerik üzerinden tartışılmıştır: doğru düşünce nedir, hakikat nasıl elde edilir, akıl nasıl işler? Oysa mühendislik bakışı farklı bir sorudan başlar: bir sistem nasıl çalışır, hangi bileşenlerden oluşur, hangi kısıtlar altında hangi çıktıları üretir ve bu çıktılar nasıl optimize edilebilir?

Bu bakış açısı zihne uygulandığında radikal bir sonuç ortaya çıkar: düşünme artık bir “içsel ışık” değil, katmanlı bir işlem sistemidir. Karpathy’nin çerçevesinde bu sistem, yalnızca teorik bir soyutlama değil, doğrudan çalıştırılan bir üretim döngüsüdür. Modelden ne istediğinin net tanımı (spesifikasyon) yazılır, model çalışır, çıktı üretilir ve doğrulama mekanizmaları devreye girer. Böylece düşünme, zihnin içinde olup biten belirsiz bir süreç olmaktan çıkar ve açıkça tanımlanabilir bir iş akışına dönüşür.

Bu sistemin girdileri vardır (algı, veri, deneyim), içsel işlem süreçleri vardır (bellek, dikkat, ilişkilendirme) ve çıktıları vardır (karar, dil, davranış). Bu üçlü yapı ilk bakışta basit görünse de, aslında modern yapay zekâdan bilişsel psikolojiye kadar uzanan geniş bir alanın ortak temel şemasını oluşturur. Dolayısıyla düşünmenin mühendislik olarak yeniden tanımı, yalnızca felsefi bir yeniden çerçeveleme değil, aynı zamanda disiplinler arası bir birleşme noktasına işaret eder.

Bu noktada tarihsel bir kırılma belirir: düşünme artık yalnızca açıklanan bir şey değildir, aynı zamanda modellenen, simüle edilen ve giderek inşa edilen bir şeydir. Karpathy’nin yaklaşımı bu dönüşümün güncel bir ifadesidir; çünkü burada model yalnızca cevap üreten bir araç değil, aynı zamanda düşünme sürecinin kendisini görünür ve kontrol edilebilir hale getiren bir sistemdir. Böylece zihne dair iki farklı yaklaşım karşı karşıya gelir. Birinci yaklaşım zihni anlamaya çalışır; ikinci yaklaşım ise zihni yeniden üretmeye ve çalıştırmaya çalışır. Bu kitap ikinci yaklaşımın içinden yazılmaktadır.

Mühendislik bakışı, düşünmeyi soyut bir varlık olmaktan çıkarıp bileşenlerine ayrılabilir bir yapı haline getirir. Bu yapı içinde artık “düşünme” tek bir süreç değildir; aksine çoklu alt süreçlerin koordinasyonudur. *Karpathy’nin çerçevesi bu koordinasyonu açık hale getirir: modelden ne istediğinin net tanımı (spesifikasyon) neyin üretileceğini tanımlar, doğrulama üretileni sürekli*

test eder, çalışma zamanı ise bu iki katmanı canlı bir etkileşim içinde tutar. Böylece düşünme, tek bir merkezden yayılan bir akış değil, sürekli geri beslenen bir üretim sistemi haline gelir.

Bu dönüşümün en önemli sonucu şudur: eğer düşünme bir sistem ise, o zaman tasarlanabilir de olabilir. İşte kitabın ikinci temel kavramı burada ortaya çıkar: zihni tasarlama problemi. Bu problem, insan zihninin yalnızca nasıl çalıştığını anlamayı değil, aynı zamanda alternatif çalışma biçimlerinin mümkün olup olmadığını sorgular. Karpathy'nin yaklaşımı bu soruya pratik bir yanıt önerir: düşünme, doğru katmanlara ayrıldığında yeniden organize edilebilir bir üretim sürecidir.

Zihni tasarlama problemi şu varsayımı içerir: eğer düşünme belirli yapısal bileşenlerden oluşuyorsa, bu bileşenler yeniden düzenlenebilir, genişletilebilir veya farklı mimariler içinde yeniden kurulabilir. Bu durumda zihin, doğanın sabit bir ürünü olmaktan çıkar; mühendislik tasarımına açık bir sistem haline gelir. Bu bakış açısı, insan zihnini bir "veri verilen yapı" olarak değil, bir "tasarım uzayı" olarak görür.

Bu noktada önemli bir ayrım ortaya çıkar: açıklamak ile tasarlamak aynı şey değildir. Açıklama, mevcut bir sistemin nasıl çalıştığını anlamaya çalışır. Tasarım ise o sistemin nasıl çalışması gerektiğini yeniden kurar. Karpathy'nin çerçevesi, açıklamadan tasarıma geçişi hızlandıran bir ara katman gibi çalışır: düşünme sürecini parçalayarak yeniden düzenlenebilir hale getirir.

Ancak bu dönüşüm yalnızca teknik değildir. Aynı zamanda felsefi bir sonuç doğurur: düşünme artık insan zihninin tekeline ait bir fenomen değildir. Model sistemleri, yazılım yapıları ve üretim döngüleri de benzer düşünme biçimleri sergileyebilir. Bu durum, düşünmenin tanımını genişletir ve onu bireysel bir iç süreç olmaktan çıkararak sistemsel bir özellik haline getirir.

İşte bu kitap tam olarak bu soruya odaklanır: düşünme nasıl bir mimari olarak kurulabilir? Karpathy'nin önerdiği üretim sistemleri bu mimarinin ilk pratik örneklerinden biridir. Bu mimari hangi katmanlardan oluşur? Hangi süreçler zorunludur, hangileri türetilebilir? Ve en önemlisi, bu mimari insan zihniyle sınırlı mıdır, yoksa daha genel bir "üretim sistemi" olarak genişletilebilir mi?

Bu sorulara yaklaşmak için önce zihni bir bütün olarak değil, bir sistem olarak görmek gerekir. Sistem düşüncesi bize şunu öğretir: karmaşık yapılar, basit bileşenlerin etkileşiminden doğar. Karpathy'nin çerçevesi bu ilkenin metin üretimine uygulanmış halidir. Modelden ne istediğinin net tanımı (spesifikasyon), doğrulama ve çalışma zamanı (runtime) katmanları birlikte çalışarak düşünmeyi yeniden yapılandırır.

Bu nedenle giriş bölümünün temel sonucu şudur: düşünme artık bir "iç dünya" değildir; bir "mimari problem"dir. Karpathy'nin önerdiği gibi, bu mimari artık yalnızca açıklanan değil, çalıştırılan ve tasarlanan bir sistemdir. Ve her mimari problem gibi, o da analiz edilebilir, modellenebilir ve yeniden tasarlanabilir.

Bu çerçeveyi daha geniş bir bağlama oturttuğumuzda iki önemli kavram belirginleşir: Yazılım 2.0 ve kendini iyileştiren yapay zekâ sistemleri. Bu iki kavram, Karpathy'nin önerdiği düşünme

mimarisinin yalnızca bir yorumu değil, aynı zamanda onun modern yapay zekâ mühendisliğindeki karşılığıdır.

Andrej Karpathy'nin ortaya koyduğu Yazılım 2.0 yaklaşımı, yazılımın doğasını kökten değiştirir. Klasik Yazılım 1.0'da sistem davranışı doğrudan insan tarafından yazılan kurallar, fonksiyonlar ve kontrol akışları ile belirlenir. Yani mühendis, sistemi satır satır inşa eder. Buna karşılık Yazılım 2.0'da davranış, açıkça yazılmış kurallardan değil, veriden öğrenilen parametrelerden ortaya çıkar. Kod artık davranışı doğrudan tanımlamaz; yalnızca öğrenme sürecini ve çerçeveyi tanımlar. Böylece yazılım, "yazılan bir şey" olmaktan çıkar, "eğitilen bir sistem" haline gelir.

Bu dönüşümün en kritik sonucu şudur: kontrol noktası koddan modele kayar. Artık sistemin ne yapacağı, büyük ölçüde veri, optimizasyon süreci ve geri besleme mekanizmaları tarafından belirlenir. Bu durum, Karpathy'nin önerdiği belirtme–doğrulama–çalışma zamanı ayrımıyla doğrudan örtüşür. Çünkü Yazılım 2.0'da "belirtme" açık kod değil, veri ve hedef fonksiyondur; "doğrulama" performans ölçümü ve değerlendirme süreçleridir; "çalışma zamanı" ise modelin gerçek dünyadaki davranışıdır.

Bu noktada ikinci kavram devreye girer: kendini iyileştiren yapay zekâ sistemleri. Bu sistemler, yalnızca bir kez eğitilen ve sabit kalan modeller değildir. Aksine, kendi çıktısını değerlendiren, hatalarını analiz eden ve yeniden üretim döngüsüne giren yapılardır. Bu yaklaşımda sistem, dışarıdan sürekli müdahale gerektiren pasif bir araç olmaktan çıkar; kendi iç geri besleme mekanizmasıyla gelişen aktif bir üretim organizmasına dönüşür.

Bu tür sistemlerin temel özelliği, üretim ve değerlendirme süreçlerinin ayrılmamasıdır. Model bir çıktı üretir, bu çıktı başka bir değerlendirme katmanı tarafından incelenir, gerektiğinde düzeltilir veya yeniden yazılır. Bu döngü yalnızca insan geri bildirimiyle (RLHF gibi) değil, aynı zamanda otomatik değerlendirme modelleriyle de desteklenebilir. Böylece sistem, zaman içinde kendi hatalarını daha iyi tanıyabilen ve üretim kalitesini artırabilen bir yapıya evrilir.

Bu iki kavram birlikte düşünüldüğünde ortaya daha geniş bir çerçeve çıkar: düşünme artık yalnızca insan zihnine ait bir süreç değil, aynı zamanda öğrenebilen, geri besleme ile kendini düzenleyebilen sistemlerin ortak özelliği haline gelir. Yazılım 2.0 bu sürecin yapısal temelini oluştururken, kendini iyileştiren sistemler onun dinamik davranış biçimini temsil eder.

Anthropic gibi modern yapay zekâ sistemlerinde görülen mimari de bu doğrultudadır. Claude gibi modeller yalnızca tek adımda cevap üreten sistemler değil, aynı zamanda kendi çıktısını değerlendiren ve gerektiğinde yeniden üreten döngüsel yapılardır. Bu, Karpathy'nin çerçevesinin yalnızca teorik bir düşünce modeli olmadığını, doğrudan modern yapay zekâ mühendisliğinin operasyonel bir gerçeğine dönüştüğünü gösterir.

Sonuç olarak bu iki kavram, girişte tarif edilen düşünme mimarisini daha teknik bir zemine taşır. Düşünme artık yalnızca bir iç süreç değil, veriyle şekillenen (Yazılım 2.0), geri besleme ile gelişen (self-improving systems) ve çalışma anında yeniden organize olan bir üretim sistemidir. Bu bakış açısı, zihni açıklanacak bir fenomen olmaktan çıkarıp, tasarlanabilir ve optimize edilebilir bir mimari nesne haline getirir.

Andrej Karpathy Kimdir?

Andrej Karpathy, çağdaş yapay zekâ tarihinin en önemli figürlerinden biri olarak, yalnızca bir araştırmacı ya da mühendis değil; aynı zamanda modern *"düşünme sistemleri"*nin nasıl inşa edileceğini yeniden tanımlayan bir mimar olarak görülebilir. Onu anlamak, tek bir biyografi okumaktan çok, yapay zekânın son on yıl içinde geçirdiği zihinsel ve teknik dönüşümü okumak anlamına gelir. *Çünkü Karpathy'nin hikâyesi, bireysel bir kariyerden çok daha fazlasıdır: yazılımın koddan öğrenmeye, öğrenmeden ise kendi kendini düzenleyen sistemlere evrilişinin hikâyesidir.*

Karpathy 1986 yılında Slovakya'da doğar ve genç yaşta Kanada'ya taşınır. Bu coğrafi hareketlilik, onun düşünsel dünyasında da bir tür "sınır aşımı" yaratır. Toronto Üniversitesi'nde bilgisayar bilimi eğitimi alırken, özellikle Geoffrey Hinton'ın laboratuvarında çalışması, onun kaderini belirleyen en kritik eşik olur. Hinton, o dönemde derin öğrenmenin henüz marjinal kabul edildiği bir alanda çalışmaktadır. Yapay sinir ağlarının yeniden yükselişe geçeceği henüz net değildir. Ancak Karpathy, bu düşük yoğunluklu ama yüksek potansiyelli alanın içine erken girer.

Doktora çalışmalarını Stanford Üniversitesi'nde Fei-Fei Li'nin danışmanlığında tamamlar. Bu dönem, onun bilgisayarlı görü (computer vision) alanında derinleştiği ve özellikle ImageNet devriminin zihinsel altyapısını şekillendirdiği yıllardır. Görsel verinin büyük ölçekli öğrenme sistemlerine nasıl dönüştürülebileceği sorusu, onun temel araştırma eksenini oluşturur. Burada önemli olan sadece teknik başarı değil, bir fikrin doğuşudur: "veri arttıkça algoritma değil, temsil yapısı değişir."

Bu fikir, daha sonra tüm yapay zekâ paradigmasını dönüştürecek olan yaklaşımın temelidir. Çünkü Karpathy'ye göre zekâ, önceden yazılmış kuralların uygulanması değil, veriden türeyen bir temsil mühendisliğidir. Bu bakış açısı, klasik yazılım mühendisliğinin sınırlarını aşar ve yeni bir kavramı doğurur: Software 2.0.

Karpathy'nin en bilinen katkılarından biri, "Yazılım 2.0" kavramıdır. *Bu kavram ilk bakışta basit bir metafor gibi görünür: yazılım artık insanlar tarafından yazılan kodlardan değil, sinir ağlarının öğrendiği ağırlıklardan oluşur.* Ancak bu ifade, aslında çok daha derin bir epistemolojik dönüşümü işaret eder.

Geleneksel yazılımda insan, sistemin davranışını açık kurallarla tanımlar. "Eğer A olursa B yap" şeklinde ilerleyen bu yapı, deterministik bir dünya varsayar. Oysa gerçek dünya bu kadar temiz değildir. Görüntü tanıma, doğal dil işleme veya otonom karar sistemleri gibi alanlarda kuralların sayısı o kadar artar ki, sistem artık manuel olarak yazılamaz hale gelir.

Karpathy'nin önerdiği yaklaşımda ise yazılımcı artık doğrudan davranış yazmaz. Bunun yerine veri toplar, bir model tanımlar ve sistemi eğitir. Davranış, koddan değil, veriden ortaya çıkar. Bu, yazılımın doğasını değiştiren radikal bir kopuştur. *Artık yazılım bir ürün değil, öğrenen bir organizmadır.*

Bu dönüşüm, yalnızca teknik değil, aynı zamanda felsefidir. Çünkü kontrol artık insanın yazdığı satırlarda değil, sistemin içsel temsil uzayında gerçekleşir. İnsan, sistemin davranışını doğrudan değil, dolaylı olarak şekillendirir.

Karpathy'nin Tesla'daki çalışmaları, onun fikirlerinin gerçek dünyadaki en güçlü test alanlarından biri olur. Tesla'nın otonom sürüş sistemi, yalnızca bir mühendislik problemi değil, aynı zamanda dev bir öğrenme sistemi problemidir.

Burada Karpathy'nin rolü, büyük ölçekli görsel verinin gerçek zamanlı öğrenmeye nasıl dönüştürüleceğini tasarlamaktır. Milyonlarca kilometrelik sürüş verisi, sistemin öğrenme yakıtıdır. Bu veriden anlamlı davranış üretmek ise klasik yazılım yaklaşımıyla mümkün değildir.

Bu noktada Yazılım 2.0 yaklaşımı doğrudan devreye girer. Sistem, sürüş kurallarını tek tek kodlamak yerine, veriden sürüş davranışını öğrenir. Trafik ışıkları, yaya hareketleri, şerit değişimleri gibi karmaşık durumlar, istatistiksel temsil uzayında çözülür.

Karpathy'nin Tesla'daki çalışmaları, yapay zekânın artık laboratuvar dışına çıktığını ve fiziksel dünyaya entegre olduğunu gösterir. Bu, yapay zekâ tarihinde önemli bir kırılma noktasıdır: model artık sadece "anlayan" değil, "hareket eden" bir sistemdir.

Karpathy daha sonra OpenAI'ye katılır ve burada dil modelleri ve büyük ölçekli yapay zekâ sistemleri üzerine çalışır. Bu dönem, onun düşünsel evriminde yeni bir aşamadır: görsel dünyadan dilsel dünyaya geçiş.

Dil modelleri, aslında yalnızca metin üreten sistemler değildir. Onlar aynı zamanda düşünme biçimlerini taklit eden yapılar olarak da görülebilir. Karpathy'nin burada katkısı, bu sistemlerin yalnızca büyütülmesi değil, aynı zamanda davranışsal olarak anlaşılabilir hale getirilmesidir.

Bu dönemde ortaya çıkan temel fikirlerden biri şudur: model sadece çıktı üretmez, aynı zamanda bir "düşünme akışı" simüle eder. Akıl Yürütme Zinciri (Chain-of-thought) gibi yaklaşımlar, bu zihinsel simülasyonun ilk örnekleridir.

Karpathy'nin yaklaşımı, modelleri daha şeffaf ve daha kontrol edilebilir hale getirme çabasıdır. Çünkü sistem büyüdükçe, davranışın açıklanabilirliği azalır. Bu durum ise, yapay zekânın en temel problemlerinden biridir.

Karpathy'nin ismiyle anılan en önemli kavramlardan biri "Karpathy Döngüsü"dür. Bu döngü, modern yapay zekâ sistemlerinin nasıl sürekli geliştiğini açıklayan bir çerçevedir: üretim, değerlendirme, geri besleme ve yeniden üretim.

Bu yapı aslında oldukça basit görünür. Sistem bir çıktı üretir, bu çıktı test edilir, hatalar belirlenir ve sistem kendini düzeltir. Ancak bu basit döngü, doğru ölçeklendirildiğinde son derece güçlü bir davranış ortaya çıkarır: sistem kendi kendini iyileştirmeye başlar. Bu yaklaşım, yazılımı statik bir varlık olmaktan çıkarır. Artık yazılım bir kez yazılıp çalıştırılan bir şey değildir; sürekli gelişen bir süreçtir.

Anayasal Yapay Zekâ (Constitutional AI), yapay zekâ sistemlerinin davranışlarını yönlendirmek için açıkça tanımlanmış bir kurallar, ilkeler ve değerler kümesinin kullanılması yaklaşımıdır. Kavram özellikle Anthropic tarafından geliştirilmiş ve yaygınlaştırılmıştır. Anthropic gibi şirketlerin kullandığı "Anayasal YZ" ve benzeri yöntemler, bu döngünün daha güvenli ve kontrollü versiyonlarını oluşturur. Burada amaç yalnızca doğru cevap üretmek değil, aynı zamanda doğru şekilde öğrenmektir. Karpathy'nin en önemli etkisi, zekâ kavramını yeniden tanımlamasıdır. Zekâ artık yalnızca problem çözme yeteneği değil, kendi kendini optimize edebilme kapasitesidir. Bu, klasik yapay zekâ tanımlarının ötesine geçen bir yaklaşımdır.

Bu bakış açısı, aynı zamanda insan zekâsını da yeniden düşünmemize neden olur. Çünkü insan zihni de sürekli geri besleme ile çalışan bir sistemdir. Öğrenme, hatalar ve düzeltmeler üzerinden ilerler. Bu benzerlik, insan ve makine arasındaki sınırın giderek bulanıklaştığını gösterir.

Andrej Karpathy, yalnızca bir araştırmacı değildir. O, yazılımın ve zekânın nasıl tanımlandığını değiştiren bir düşünce hattının temsilcisidir. Onun çalışmaları, yapay zekânın geleceğini anlamak için bir biyografiden çok bir harita gibidir.

Bugün geldiğimiz noktada yapay zekâ sistemleri artık sadece araçlar değildir. Onlar öğrenen, kendini düzelten ve sürekli evrilen yapılardır. Bu dönüşümün arkasında ise Karpathy'nin fikirlerinin belirleyici bir etkisi vardır.

Dolayısıyla "Andrej Karpathy kimdir?" sorusunun cevabı basit değildir. O, bir kişi olduğu kadar bir paradigma değişimidir. Yazılımın, öğrenmenin ve düşünmenin yeniden tasarlanmasının adı haline gelmiştir.

Kaynaklar

Anthropic. (2023). *Claude and constitutional AI system card*. https://www.anthropic.com

Aristotle. (2009). *Prior analytics* (R. Smith, Trans.). Hackett Publishing. (Original work published ca. 350 BCE)

Ashby, W. R. (1956). *An introduction to cybernetics*. Chapman & Hall.

Bai, Y., et al. (2022). *Constitutional AI: Harmlessness from AI feedback*. https://arxiv.org/abs/2212.08073

Bohn, D. (2021). *Tesla's self-driving system explained*. The Verge. https://www.theverge.com

Boole, G. (1854). *An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities*. Walton and Maberly.

Brown, T. B., et al. (2020). *Language models are few-shot learners*. https://arxiv.org/abs/2005.14165

Chomsky, N. (1957). *Syntactic structures*. Mouton.

Descartes, R. (1998). *Discourse on method and meditations on first philosophy* (D. A. Cress, Trans.). Hackett Publishing. (Original works published 1637, 1641)

Frege, G. (1980). *Philosophical and mathematical correspondence*. University of Chicago Press.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. https://www.deeplearningbook.org

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society* (pp. 1–12). Cognitive Science Society.

Hinton, G. E., et al. (2012). *Deep neural networks for acoustic modeling*. IEEE Signal Processing Magazine. https://ieeexplore.ieee.org

Karpathy, A. (2017). *Software 2.0*. https://karpathy.medium.com/software-2-0-a64152b37c35

Karpathy, A. (2023). *Intro to large language models*. YouTube. https://www.youtube.com/watch?v=zjkBMFhNj_g

Karpathy, A. (2023). *State of GPT*. YouTube. https://www.youtube.com/watch?v=bZQun8Y4L2A

Karpathy, A. (2024). *Neural networks: Zero to hero series*. https://github.com/karpathy/nn-zero-to-hero

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

Levine, S. (2018). *End-to-end learning for self-driving cars*. https://arxiv.org/abs/1604.07316

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63*(2), 81–97. https://doi.org/10.1037/h0043158

Minsky, M. (1986). *The society of mind*. Simon & Schuster.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Prentice-Hall.

Newell, A., & Simon, H. A. (1976). Computer science as empirical inquiry. *Communications of the ACM, 19*(3), 113–126. https://doi.org/10.1145/360018.360022

Neisser, U. (1967). *Cognitive psychology*. Appleton-Century-Crofts.

OpenAI. (2023). *GPT-4 technical report*. https://arxiv.org/abs/2303.08774

Ouyang, L., et al. (2022). *Training language models to follow instructions with human feedback*. https://arxiv.org/abs/2203.02155

Russell, B., & Whitehead, A. N. (1910–1913). *Principia mathematica* (Vols. 1–3). Cambridge University Press.

Shinn, N., et al. (2023). *Reflexion: Language agents with verbal reinforcement learning*. https://arxiv.org/abs/2303.11366

Simon, H. A. (1957). *Models of man: Social and rational*. Wiley.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, 42*(1), 230–265. https://doi.org/10.1112/plms/s2-42.1.230

Turing, A. M. (1950). Computing machinery and intelligence. *Mind, 59*(236), 433–460. https://doi.org/10.1093/mind/LIX.236.433

Wang, X., et al. (2023). *Self-refine: Iterative refinement with LLMs*. https://arxiv.org/abs/2303.17651

Wiener, N. (1948). *Cybernetics: Or control and communication in the animal and the machine*. MIT Press.

Yao, S., et al. (2023). *ReAct: Synergizing reasoning and acting in language models*. https://arxiv.org/abs/2210.03629

1. TARİHSEL TEMELLER: DÜŞÜNMEYİ İNŞA ETMEK

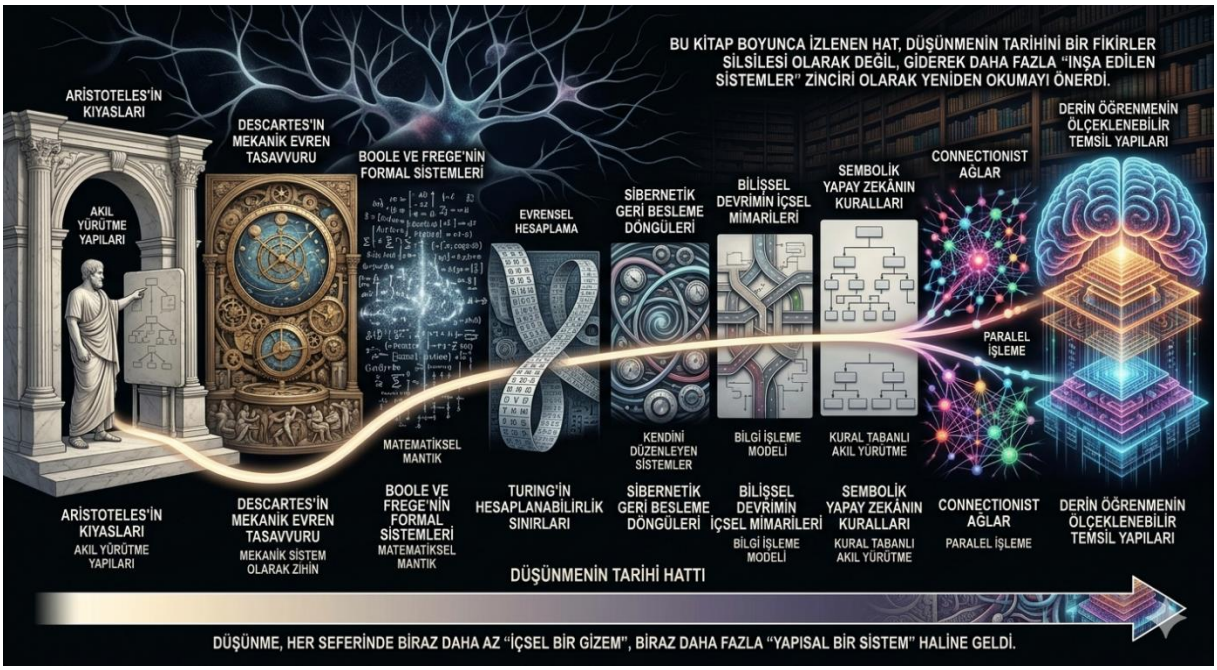
Düşünmenin bir "insan yetisi" olmaktan çıkıp bir "mimari problem" haline gelmesi, tek bir dönüm noktasının sonucu değildir. Bu dönüşüm, yüzyıllar boyunca katman katman biriken fikirlerin, birbirini dönüştürerek ilerlemesinin sonucudur. Bugün yapay zekâ sistemlerini ya da bilişsel modelleri konuşurken kullandığımız dil—katmanlar, temsiller, işlem süreçleri, hesaplanabilirlik—aslında çok uzun bir entelektüel inşanın geç ürünleridir. Bu bölüm, düşünmenin nasıl giderek "inşa edilebilir" bir yapıya dönüştüğünü, Aristoteles'ten Turing'e uzanan çizgi içinde, kesintisiz bir dönüşüm olarak ele alır.

Aristoteles ile birlikte düşünme ilk kez sistematik bir forma kavuşur. Ondan önce düşünme, daha çok kozmosun anlaşılması, varlığın sezgisel kavranışı ya da retorik bir ikna pratiği olarak görülüyordu. *Aristoteles ise düşünmeyi içerikten bağımsızlaştırarak yapısal bir zemine taşır. Onun kıyas sistemi, insan aklının nasıl çalıştığını açıklamaktan ziyade, aklın nasıl "geçerli çıkarımlar" ürettiğini tanımlar. Bu ayrım kritik bir eşiktir; çünkü burada düşünme ilk kez bir tür "işleyiş kuralları" olan sistem olarak görülmeye başlanır. Artık önemli olan ne düşünüldüğü değil, nasıl düşünüldüğüdür. Bu, düşünmenin tarihindeki ilk büyük mimari hamledir: içerikten yapıya geçiş.*

Bu yapısal düşünme fikri, yüzyıllar sonra Descartes ile birlikte farklı bir boyuta taşınır. Descartes'ın yaşadığı dönem, evrenin mekanik yasalarla açıklanmaya başlandığı bir dönemdir

ve bu dönüşüm zihni de kaçınılmaz olarak etkiler. Eğer doğa matematiksel ve mekanik yasalarla işliyorsa, zihnin de bu sistemin dışında kalması düşünülemez. Descartes, zihni bedenin dışında konumlandırarak onu ayrı bir töz olarak tanımlasa da, aynı zamanda mekanik açıklama fikrini zihinsel süreçlere doğru genişletir. Bu genişleme, düşünmenin artık yalnızca mantıksal bir yapı değil, aynı zamanda doğa içinde yer alan bir sistem olabileceği fikrini doğurur. Zihin, bu noktadan itibaren metafizik bir gizem olmaktan çıkıp açıklanabilir bir yapı haline gelmeye başlar.

Şekil 1. Düşünmenin Tarihi Hattı



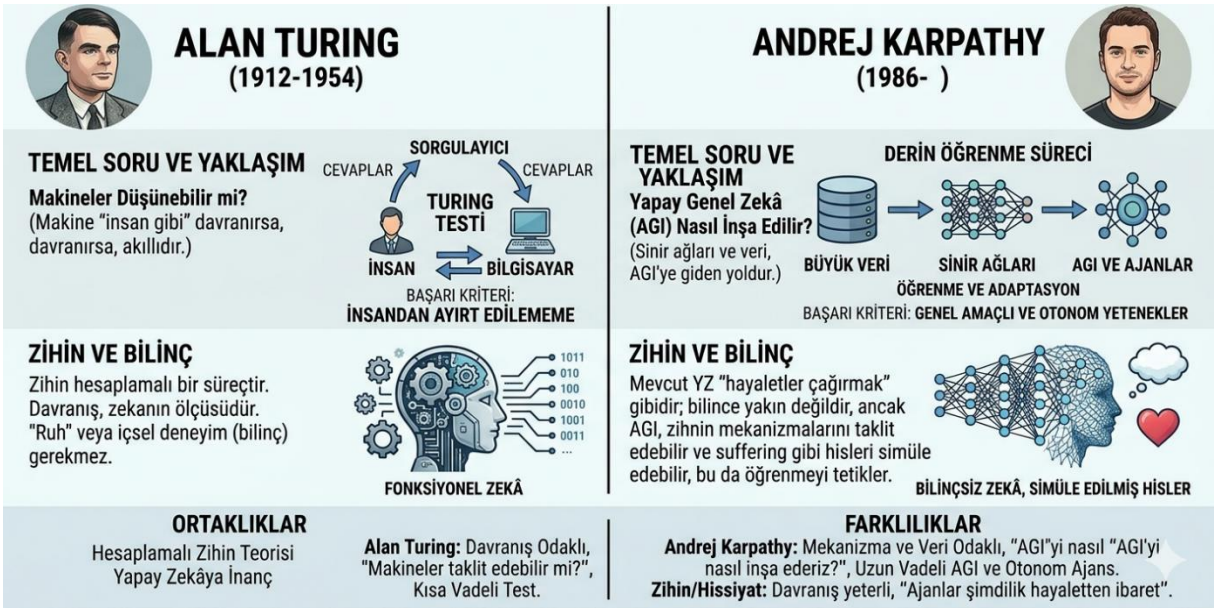
19. yüzyıla gelindiğinde bu açıklanabilirlik arzusu matematiksel bir kesinliğe dönüşür. *George Boole, mantığı cebirsel bir sistem olarak yeniden kurduğunda, düşünmenin dili değişir. Artık "doğru" ve "yanlış" gibi kavramlar, sembolik işlemlere dönüştürülebilir hale gelir.* Bu dönüşüm basit bir teknik ilerleme değildir; düşünmenin kendisinin matematiksel bir nesneye dönüşmesidir. Boole'un açtığı bu kapı, Frege ile birlikte daha da derinleşir. Frege, matematiği mantıksal temellere indirgerken, anlam ile referans arasındaki ayrımı kurarak dilin düşünme içindeki rolünü yeniden tanımlar. Russell ve Whitehead'in "Principia Mathematica" projesi ise bu çizgiyi zirveye taşır: artık hedef, tüm matematiği saf mantıksal sistemlerden türetmektir. Bu girişim tamamlanmamış olsa da, bıraktığı miras nettir: düşünme artık tamamen formal bir sistem olarak ele alınabilir.

Bu formalizasyon sürecinin doğal bir devamı Alan Turing ile gelir ve burada kritik kırılma aslında sadece "hesaplama" fikrinin ortaya çıkması değildir; düşünmenin, ilk kez tamamen mekanik ve kurallara indirgenebilir bir süreç olarak tanımlanmasıdır. *Turing, hesaplanabilirlik kavramını tanımlarken şunu gösterir: görünürde son derece karmaşık olan akıl yürütme süreçleri, yeterince temel işlemlere indirgenirse adım adım çalışan bir makine tarafından üretilebilir.* Turing makinesi bu anlamda yalnızca teorik bir cihaz değil, düşünmenin "parçalanabilirlik

sınırı"dır. Yani bir şey Turing makinesiyle ifade edilemiyorsa, o şey artık klasik anlamda "hesaplanabilir düşünme" alanının dışında kalır.

Bu yaklaşım, zihni iki radikal şekilde yeniden konumlandırır. Birincisi, düşünme artık içsel bir sezgi veya açıklanamaz bir bilinç akışı değil, açıkça tanımlanabilir sembolik işlemler zinciri olarak görülür. İkincisi, bu işlemler zinciri doğru şekilde organize edilirse, insan zihninin ürettiği davranışlara denk çıktılar üretmek mümkün hale gelir. Bu noktada zihin, metafizik bir "öz" olmaktan çıkar; giriş, işlem ve çıkışları olan bir sistem haline gelir.

Şekil 2. Alan Turing ve Andrej Karpathy'nin Düşünceleri



Karpathy'nin düşünme sistemlerine yaklaşımı tam olarak bu Turingci çizginin modern bir devamı gibi okunabilir, ancak önemli bir farkla: Turing daha çok "ne hesaplanabilir?" sorusuna odaklanırken, Karpathy "hesaplama nasıl organize edilir ve çalıştırılır?" sorusuna gider. Turing'in açtığı kapı, zihnin prensipte algoritmik olduğunu söylerken; Karpathy'nin yaklaşımı, bu algoritmik yapının pratikte nasıl bir üretim iş akışına dönüştürülebileceğini gösterir. Özellikle metin üretim sisteminde önerdiği belirtme–doğrulama–çalışma zamanı ayrımı, Turing'in soyut hesaplama modelini somut bir çalışma mimarisine çevirir.

Alan Turing'in iki temel katkısı bu tartışmanın arka planını oluşturur: Turing makinesi ve taklit oyunu (imitation game). Turing makinesi, son derece basit sembolik işlemlerle çalışan soyut bir hesaplama modelidir ve şunu gösterir: yeterince temel adımlara indirgenen her işlem, prensipte bir makine tarafından gerçekleştirilebilir. Bu, düşünmenin de teorik olarak algoritmik bir yapıya indirgenebileceği fikrinin temelini oluşturur. Yani karmaşık görünen zihinsel süreçler, uygun şekilde parçalandığında hesaplanabilir işlemler zinciri olarak yeniden yazılabilir.

Taklit oyunu ise bu fikri farklı bir düzleme taşır. Burada amaç, bir makinenin insan gibi düşünüp düşünmediğini "içsel olarak" anlamak değildir. Bunun yerine, makinenin dışarıdan bakıldığında insanla ayırt edilemeyecek kadar benzer cevaplar üretip üretemediği test edilir. Başka bir

deyişle, düşünmenin iç mekanizması değil, davranışsal çıktısı ölçüt haline gelir. Bu yaklaşım, zihni içsel bir bilinç problemi olmaktan çıkarıp gözlemlenebilir bir performans problemi haline getirir.

Bu noktada bağlantı daha net hale gelir: Turing makinesi, düşünmeyi teorik olarak "indirgenebilir" ilan eder; Karpathy'nin çerçevesi ise bu indirgenmiş düşünmeyi günlük kullanımda yönetilebilir bir süreç haline getirir. Yani Turing bize "zihin bir hesaplama olabilir" derken, Karpathy "o hesaplamayı nasıl katmanlara ayırıp kontrollü bir üretim sistemine dönüştürürüz?" sorusunu cevaplar.

Turing'in taklit oyunu bu dönüşümün ikinci kritik aşamasını temsil eder. Burada artık önemli olan, sistemin nasıl "düşündüğü" değil, dışarıdan bakıldığında insan benzeri bir davranış üretip üretemediğidir. Bu, zihni içsel bir fenomen olmaktan çıkarıp tamamen davranışsal olarak test edilebilir bir sistem haline getirir. Böylece değerlendirme ölçütü "iç mekanizma" değil, "çıktı kalitesi ve tutarlılığı" olur.

Karpathy'nin yaklaşımı bu çizgiyi daha da ileri taşır: yalnızca insan benzeri çıktı üretmek değil, bu çıktının hangi belirtme ile üretildiğini, hangi doğrulama süreçlerinden geçtiğini ve hangi çalışma zamanı dinamikleriyle ortaya çıktığını görünür hale getirir. Böylece Turing'in davranış düzeyinde bıraktığı soru, Karpathy'de tamamen bir mühendislik ve mimari tasarım problemine dönüşür.

Karpathy'nin yaklaşımı bu davranışsal bakışı bir adım daha ileri taşır. Sadece çıktıyı değil, çıktıya giden süreci de görünür ve kontrol edilebilir hale getirir. Yani Turing'in davranış düzeyinde bıraktığı "taklit" problemi, Karpathy'de bir "üretim mimarisi" problemine dönüşür. Model artık yalnızca taklit eden bir sistem değil, belirli bir belirtim üzerinden sürekli doğrulanan ve çalışma zamanı içinde optimize edilen bir üretim hattıdır.

Bu çerçevede zihin, üç katmanda yeniden okunur: Turing düzeyinde hesaplanabilirlik, Karpathy düzeyinde üretim mimarisi, modern yapay zekâ düzeyinde ise sürekli çalışan bir sistem. Böylece düşünme, artık yalnızca "akıl yürütme" değil; tasarlanabilir, çalıştırılabilir ve geri beslenebilir bir hesaplama üretim sistemi haline gelir.

Bu tarihsel çizgi, geriye dönüp bakıldığında tek bir yönelim gösterir: düşünme giderek soyut bir deneyim olmaktan çıkarak yapısal, formal ve nihayet hesaplanabilir bir nesneye dönüşmektedir. Aristoteles'in kıyaslarından Turing'in makinelerine uzanan bu hat, aslında tek bir fikrin farklı evreleridir: zihni bir sistem olarak tanımlama fikri.

Ancak bu dönüşüm yalnızca mantık ve matematik düzeyinde gerçekleşmez. 20. yüzyılın ortasında Norbert Wiener ile birlikte yeni bir paradigma ortaya çıkar: sibernetik. Wiener, geri besleme mekanizmaları üzerinden hem canlı sistemleri hem de makineleri açıklamaya çalışırken, düşünmenin de bir kontrol sistemi olarak ele alınabileceğini gösterir. Bu bakış açısı, zihni yalnızca sembolik işlemler bütünü olarak değil, aynı zamanda çevreyle sürekli etkileşim halinde olan dinamik bir sistem olarak görmeye başlar. Ross Ashby'nin çeşitlilik yasası ise bu

çerçeveyi daha da derinleştirir; karmaşıklıkla baş edebilmek için sistemin içsel çeşitliliğinin artırılması gerektiğini savunur. Bu fikir, daha sonra bilişsel sistemlerin tasarımında kritik bir rol oynayacaktır.

Bu noktada düşünme artık yalnızca içsel bir hesaplama süreci değil, aynı zamanda çevresel geri beslemelerle şekillenen bir sistem davranışıdır. Zihin, kapalı bir yapı değil, açık bir sistemdir. Bu açık sistem fikri, daha sonra hem bilişsel bilimlerde hem de yapay zekâ araştırmalarında merkezi bir rol oynayacaktır.

1950'lerden itibaren bu fikirler daha somut bir mühendislik programına dönüşür. Allen Newell ve Herbert Simon, insan problem çözme süreçlerini algoritmik olarak modellemeye çalışırken, zihni doğrudan bir "problem çözücü sistem" olarak tanımlarlar. General Problem Solver (Genel Problem Çözücü) gibi sistemler, insan düşüncesini adım adım simüle etmeye çalışır. Burada kritik dönüşüm şudur: düşünme artık yalnızca açıklanan bir şey değil, yeniden üretilen bir süreçtir. Simon'un "bounded rationality" (sınırlı rasyonalite) kavramı ise bu modeli daha gerçekçi hale getirir; insan zihni sınırsız bir optimizasyon makinesi değildir, aksine sınırlı kaynaklar içinde tatmin edici çözümler üreten bir sistemdir. Bu, düşünmenin mühendislik açısından modellenebilirliğini güçlendirir.

Marvin Minsky'nin çalışmaları bu çizgiyi Turing'in hesaplanabilirlik fikrinden alıp çok daha yapısal ve mimari bir seviyeye taşır. Minsky, özellikle "The Society of Mind" (Zihin Toplumu) adlı eserinde zihni tekil, bütüncül ve merkezi bir yapı olarak değil; birbirleriyle etkileşen çok sayıda küçük, basit ve uzmanlaşmış süreçten oluşan bir sistem olarak tanımlar. Bu yaklaşımda zekâ, tek bir "merkez akıl" tarafından üretilmez. *Bunun yerine, hafıza, algı, problem çözme ve karar verme gibi farklı görevleri üstlenen mikro süreçlerin bir araya gelmesiyle ortaya çıkan bir toplumsal etkileşim sonucu oluşur. Yani zihin, tek bir varlık değil, kendi içinde sürekli pazarlık eden, iş birliği yapan ve zaman zaman çatışan alt modüllerden oluşan bir "toplum" gibidir.*

Bu fikir, düşünme kavramını radikal biçimde dağıtır. Artık "zihin nerede?" sorusunun net bir cevabı yoktur; çünkü zihin tek bir yerde değil, süreçler arasındaki etkileşim ağında ortaya çıkar. Bu durum, modern yapay zekâ sistemlerinde gördüğümüz dağıtık temsillere ve çok katmanlı öğrenme yapılarına oldukça güçlü bir öncül oluşturur. Bugün sinir ağlarında tek bir karar mekanizması yerine, farklı katmanlarda farklı temsil ve karar süreçlerinin birlikte çalışması, Minsky'nin sezgisel olarak tarif ettiği bu "çoklu ajanlı zihinsel yapı" ile güçlü bir paralellik taşır.

Minsky'nin bir diğer önemli katkısı "Frame Theory" (Çerçeve Teorisi) ile gelir. Bu teoriye göre bilgi, izole semboller halinde değil, belirli bağlamları temsil eden "çerçeveler" içinde organize edilir. *Bir çerçeve, bir durumu anlamak için gerekli olan tipik bilgi paketini içerir; örneğin "bir restoran", "bir sınıf" ya da "bir şehir caddesi" gibi bağlamlar, yalnızca nesnelerden değil, bu nesneler arasındaki ilişkilerden oluşan yapısal bir bütünlük olarak temsil edilir. Zihin, yeni bir durumla karşılaştığında bu çerçeveleri aktive eder ve bilgiyi bağlam içinde yorumlar.*

Bu iki yaklaşım birlikte düşünüldüğünde —Zihin Toplumu ve Çerçeve Teorisi— ortaya çıkan resim oldukça modern bir yapay zekâ mimarisine benzer: dağıtık süreçler, bağlamsal temsiller

ve merkezi olmayan karar yapıları. Bu, günümüz derin öğrenme sistemlerinin ve özellikle transformer tabanlı modellerin çalışma mantığıyla şaşırtıcı derecede uyumludur.

Karpathy'nin çerçevesi bu tarihsel çizgiye eklendiğinde daha da net bir süreklilik ortaya çıkar. Minsky zihni "çoklu ajanların etkileşimi" olarak görürken, Karpathy bu etkileşimi çalıştırılabilir bir üretim sistemine dönüştürür. Yani Minsky'nin teorik olarak tarif ettiği dağıtık zihin, Karpathy'de belirleme (spesifikasyon), doğrulama ve çalışma zamanı (runtime) katmanlarıyla yönetilebilir bir mühendislik mimarisine dönüşür. Bu açıdan bakıldığında, Minsky'nin fikirleri zihni parçalarına ayırırken, Karpathy bu parçaların nasıl organize edilip birlikte çalıştırılacağını gösteren bir operasyonel çerçeve sunar.

20. yüzyılın ikinci yarısında bilişsel devrim ile birlikte zihnin iç yapısı bilimsel olarak yeniden keşfedilir. George Miller'ın çalışma belleği üzerine çalışmaları, zihinsel kapasitenin sınırlarını ortaya koyar. Noam Chomsky, dilin öğrenilmiş davranışlardan ziyade doğuştan gelen yapısal bir sistem olduğunu savunarak zihnin mimari bir temel üzerine kurulu olduğunu gösterir. Ulric Neisser ise bilişsel psikolojiyi bağımsız bir bilim dalı haline getirir ve zihni içsel süreçlerin bilimsel inceleme alanı olarak yeniden kurar.

1980'lerde bağlantısallık ile birlikte bu tablo yeniden değişir. Zihin artık sembolik kurallarla değil, bağlantısal ağırlıklar ve dağıtık temsillerle açıklanmaya başlanır. *Geoffrey Hinton'un çalışmaları, öğrenmenin merkezi bir kural sisteminden ziyade, veri içinden türeyen istatistiksel yapılarla mümkün olduğunu gösterir. Bu, düşünmenin yapısal modelini kökten değiştirir: artık zihin, kuralların değil, ilişkilerin sistemidir.*

Bu tarihsel çizgi sonunda modern yapay zekâya ulaşır. Derin öğrenme ve transformer mimarileri, düşünmeyi yeniden bir üretim süreci olarak tanımlar. Artık sistemler yalnızca veri işleyen yapılar değil, bağlam içinde anlam üreten dinamik mimarilerdir. Bu noktada Karpathy'nin çerçevesi devreye girer: düşünme, üç katmanlı bir üretim sistemi olarak yeniden tanımlanır—spesifikasyon, doğrulama ve çalışma ortamı. Ancak en kritik yenilik, bu sistemin içine yerleştirilen içsel diyalog mekanizmasıdır. *Model artık yalnızca çıktı üreten bir yapı değil, kendi içinde konuşan, kendini kontrol eden ve yeniden düzenleyen bir sistemdir.*

Tüm bu tarihsel gelişim tek bir sonuca çıkar: düşünme, sabit bir öz değil, sürekli yeniden tasarlanan bir mimaridir. *Aristoteles'in mantıksal yapısından Turing'in hesaplanabilir makinelerine, Minsky'nin zihin toplumundan Karpathy'nin üretim sistemlerine kadar uzanan bu çizgi, zihni giderek daha fazla mühendislik nesnesine dönüştürmüştür. Bu nedenle bugün temel soru artık "düşünme nedir?" değil, "düşünme nasıl tasarlanır?" sorusudur.*

Kaynaklar

Aristotle. (1984). *The complete works of Aristotle: The revised Oxford translation* (J. Barnes, Ed.). Princeton University Press.

Ashby, W. R. (1956). *An introduction to cybernetics*. Chapman & Hall.

Boole, G. (1854). *An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities*. Walton and Maberly.

Chomsky, N. (1957). *Syntactic structures*. Mouton.

Descartes, R. (1998). *Discourse on method and meditations on first philosophy* (D. A. Cress, Trans.). Hackett Publishing. (Original works published 1637 and 1641)

Frege, G. (1980). *Philosophical and mathematical correspondence* (H. Kaal, Trans.). University of Chicago Press.

Frege, G. (1984). *Collected papers on mathematics, logic, and philosophy* (B. McGuinness, Ed.). Basil Blackwell.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society* (pp. 1–12). Lawrence Erlbaum Associates.

Hofstadter, D. R. (1979). *Gödel, Escher, Bach: An eternal golden braid*. Basic Books.

Karpathy, A. (2025). *Software 3.0* [Conference presentation]. AI Startup School, San Francisco, CA, United States.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63*(2), 81–97. https://doi.org/10.1037/h0043158

Minsky, M. (1974). A framework for representing knowledge. *MIT Artificial Intelligence Laboratory Memo No. 306*. Massachusetts Institute of Technology.

Minsky, M. (1986). *The society of mind*. Simon & Schuster.

Neisser, U. (1967). *Cognitive psychology*. Appleton-Century-Crofts.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Prentice-Hall.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.

Russell, B., & Whitehead, A. N. (1910–1913). *Principia mathematica* (Vols. 1–3). Cambridge University Press.

Sejnowski, T. J. (2018). *The deep learning revolution*. MIT Press.

Simon, H. A. (1957). *Models of man: Social and rational*. John Wiley & Sons.

Simon, H. A. (1996). *The sciences of the artificial* (3rd ed.). MIT Press.

Stich, S. P. (1983). *From folk psychology to cognitive science: The case against belief*. MIT Press.

Thompson, E., Rosch, E., & Varela, F. J. (1991). *The embodied mind: Cognitive science and human experience*. MIT Press.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, 42*(2), 230–265. https://doi.org/10.1112/plms/s2-42.1.230

Turing, A. M. (1937). On computable numbers, with an application to the Entscheidungsproblem: A correction. *Proceedings of the London Mathematical Society, 43*(2), 544–546. https://doi.org/10.1112/plms/s2-43.6.544

Turing, A. M. (1950). Computing machinery and intelligence. *Mind, 59*(236), 433–460. https://doi.org/10.1093/mind/LIX.236.433

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*, 5998–6008.

Wiener, N. (1948). *Cybernetics: Or control and communication in the animal and the machine*. MIT Press.

Winograd, T., & Flores, F. (1986). *Understanding computers and cognition: A new foundation for design*. Ablex Publishing.

2. SİSTEM FİKRİNİN DOĞUŞU

Düşünme tarihinin en kritik dönüşümlerinden biri, zihnin yalnızca mantıksal bir çıkarım mekanizması ya da sembolik bir işlem sistemi olarak değil, çevreyle sürekli etkileşim halinde olan dinamik bir sistem olarak ele alınmaya başlanmasıdır. Bu dönüşüm, 20. yüzyılın ortasında ortaya çıkan sibernetik düşünceyle birlikte belirginleşir ve kısa sürede hem mühendislik hem de bilişsel bilimler için ortak bir çerçeve haline gelir. Bu aşamadan itibaren zihin, kapalı bir hesaplama alanı olmaktan çıkar; geri besleme döngüleriyle şekillenen, çevreye tepki veren ve bu tepkiler üzerinden kendisini sürekli yeniden düzenleyen bir yapı olarak düşünülmeye başlanır.

Bu yeni yaklaşımın en temel varsayımı şudur: hiçbir bilişsel süreç izole değildir. Düşünme, dış dünyadan gelen girdileri işleyen ve bunlara tepki üreten kapalı bir mekanizma değil, sürekli olarak çevreyle bilgi alışverişi yapan açık bir sistemdir. Bu sistemde her çıktı, çevreyi değiştirir; çevredeki her değişim ise yeniden sisteme geri döner ve onu yeniden biçimlendirir. Böylece düşünme, tek yönlü bir hesaplama zinciri olmaktan çıkar ve sürekli kendini güncelleyen bir döngü haline gelir. Bu döngü fikri, modern sistem düşüncesinin temelini oluşturur.

Norbert Wiener'in sibernetik yaklaşımı tam da bu dönüşümün merkezinde yer alır ve aslında Turing ile Minsky arasında kalan boşluğu "dinamik davranış" düzeyinde doldurur. Wiener, başlangıçta mühendislik ve iletişim problemlerini çözmek amacıyla geliştirdiği geri besleme (feedback) kavramını, çok kısa sürede yalnızca makinelerle sınırlı olmayan evrensel bir

organizasyon ilkesine dönüştürür. *Özellikle II. Dünya Savaşı sırasında anti-hava savunma sistemleri ve otomatik hedef takip problemleri üzerinde çalışırken ortaya çıkan bu fikir, bir sistemin çevresiyle sürekli etkileşim içinde olduğunu ve bu etkileşimin davranışı şekillendirdiğini gösterir. Ancak Wiener'in en radikal adımı, bu mekanizmanın yalnızca teknik sistemlere değil, canlı organizmalara ve nihayetinde zihinsel süreçlere de uygulanabileceğini ileri sürmesidir.*

Geri besleme, en basit haliyle bir sistemin kendi çıktısını tekrar girdiye dönüştürmesi anlamına gelir. Bu döngü sayesinde sistem, yalnızca önceden belirlenmiş kuralları izleyen bir yapı olmaktan çıkar ve çevresel değişimlere tepki veren, kendini sürekli ayarlayan bir organizmaya dönüşür. Örneğin bir termostat, ortam sıcaklığını ölçer, hedef sıcaklıktan sapmayı bir hata olarak değerlendirir ve bu hatayı azaltacak şekilde ısıtma sistemini devreye sokar ya da kapatır. Bu basit mekanizma bile aslında çok daha genel bir ilkenin mikro modelidir: davranış, sabit bir planın uygulanması değil, sürekli hata düzeltme sürecidir.

Wiener bu ilkeyi daha ileri taşıyarak, benzer mekanizmaların biyolojik sistemlerde de çalıştığını gösterir. İnsan vücudu, dengeyi (homeostasis) korumak için sürekli geri besleme döngüleri kullanır; kan şekeri, vücut ısısı, denge ve motor kontrol gibi süreçler bu ilkeye dayanır. Buradan zihne geçildiğinde ise daha soyut ama daha güçlü bir sonuç ortaya çıkar: düşünme de benzer bir kontrol mekanizmasıdır. İnsan zihni, çevreden gelen bilgiyi sürekli değerlendirir, beklentilerle karşılaştırır, hata üretir ve bu hataya göre davranışlarını yeniden ayarlar. Yani zihinsel süreçler de sabit bir planın uygulanması değil, sürekli bir denge arayışıdır.

Bu bakış açısıyla zihin, artık klasik anlamda sembolleri işleyen durağan bir makine değildir. Aksine, çevreyle sürekli etkileşim halinde olan, kendi iç durumunu dış dünyaya göre güncelleyen dinamik bir kontrol sistemi haline gelir. Düşünme, bu sistem içinde tek seferlik bir hesaplama değil, sürekli tekrar eden bir döngü olarak ortaya çıkar: algıla, karşılaştır, hata üret, düzelt, yeniden dene.

Wiener'in yaklaşımı burada kritik bir kırılma yaratır çünkü düşünmeyi yalnızca "içsel hesaplama" olarak değil, zamana yayılan bir kontrol problemi olarak yeniden tanımlar. Bu, zihni statik bir yapı olmaktan çıkarıp sürekli akan bir süreç haline getirir. Artık önemli olan tek bir doğru sonuç değil, sistemin zaman içinde nasıl stabilize olduğudur.

Bu çerçeve, Karpathy'nin modern metin üretim sistemi yaklaşımıyla doğrudan bir paralellik kurar. Karpathy'nin önerdiği belirtme–doğrulama–çalışma zamanı yapısı aslında Wiener'in geri besleme fikrinin modern bir versiyonudur: belirtme hedefi tanımlar, çalışma zamanı üretimi gerçekleştirir, doğrulama ise geri besleme mekanizması olarak hatayı ölçer ve sistemi yeniden yönlendirir. Böylece metin üretimi, tek yönlü bir akış olmaktan çıkar ve sürekli kendini düzelten bir döngü haline gelir.

Bu açıdan bakıldığında Wiener'in sibernetiği yalnızca mühendislik tarihinin bir aşaması değildir; aynı zamanda düşünmenin modern mimarisinin temel taşlarından biridir. Çünkü burada zihin ilk kez açıkça bir "denge arayan sistem", yani sürekli geri besleme ile kendini organize eden bir yapı olarak tanımlanır.

Şekil 3. Norbert Wiener ile Andrej Karpathy'nin Bakış Açıları

Wiener'in bu yaklaşımında düşünme, temsil üretmekten çok denge üretme süreci olarak yeniden tanımlanır. Zihin, dış dünyanın içsel bir kopyasını oluşturmak yerine, bu dünya içinde başarılı bir şekilde davranabilmek için sürekli ayarlanan bir kontrol mekanizmasıdır. Bu nedenle sibernetik zihin, statik bilgi depolayan bir yapı değil, sürekli hareket halinde olan bir uyum sistemidir. Bu uyum, yalnızca içsel süreçlere değil, çevreden gelen sürekli geri beslemelere bağlıdır. Böylece zihin, kapalı bir hesaplama kutusu olmaktan çıkar ve açık bir etkileşim ağına dönüşür.

Bu çerçevenin daha derin bir boyutu Ross Ashby ile ortaya çıkar ve aslında sibernetik düşüncenin "kontrol" fikrinden "uyum ve zorunlu çeşitlilik" fikrine doğru genişlemesini temsil eder. Ashby, sistemlerin karmaşıklıkla nasıl başa çıkabileceği sorusunu merkeze alırken problemi yalnızca mühendislik düzeyinde değil, neredeyse varoluşsal bir sistem problemi olarak ele alır: Bir sistem, kendi sınırlarını aşan bir çevreyle karşılaştığında nasıl ayakta kalabilir? Bu soruya verdiği yanıt, sibernetiğin en temel ve en güçlü ilkelerinden birine dönüşür: bir sistem, ancak çevresinin karmaşıklığına eşdeğer bir içsel çeşitliliğe sahipse stabil kalabilir.

Bu ilke, literatürde Ashby'nin "çeşitlilik yasası" (Law of Requisite Variety) olarak bilinir ve sistem düşüncesinin en sert, en indirgenemez sonuçlarından biridir. Burada artık geri besleme tek başına yeterli bir açıklama değildir; çünkü geri besleme yalnızca hatayı düzeltme mekanizmasını açıklar, fakat sistemin neden bazı durumlarda tamamen çöktüğünü açıklamaz. *Ashby'nin katkısı tam burada devreye girer: sorun kontrol mekanizmasının varlığı değil, sistemin sahip olduğu içsel durum uzayının zenginliğidir. Yani bir sistemin çevreyi yönetebilmesi için sadece geri bildirim alması değil, aynı zamanda o geri bildirimlere karşılık verebilecek yeterli sayıda içsel davranış seçeneğine sahip olması gerekir.*

Bu fikir, sistem teorisini kökten değiştirir çünkü artık odak nokta "doğru kontrol kuralı" değil, "yeterli çeşitlilikte içsel yapı"dır. Başka bir deyişle, bir sistem ne kadar iyi tasarlanırsa tasarlansın, eğer içsel repertuarı çevrenin olası durumlarını karşılayacak kadar geniş değilse, o sistem kaçınılmaz olarak başarısız olur. Bu, mühendislikte basit bir optimizasyon problemi değil, yapısal bir sınır yasasıdır.

Ashby'nin yaklaşımında çevre her zaman sistemden daha karmaşıktır. Bu, çok kritik bir varsayımdır çünkü gerçek dünyanın tam anlamıyla modellenemeyeceğini kabul eder. Çevre sürekli değişir, beklenmedik durumlar üretir ve hiçbir kapalı model bu değişkenliği tam olarak kapsayamaz. *Bu durumda sistemin hayatta kalabilmesi için iki temel strateji ortaya çıkar. Birincisi, çevreyi aşırı basitleştirmek ve onu düşük boyutlu bir modele indirgemektir; ancak bu yaklaşım kaçınılmaz olarak kör noktalar üretir ve sistemin kırılganlığını artırır. İkincisi ise çok daha güçlü ama aynı zamanda daha maliyetli bir stratejidir: sistem kendi içsel karmaşıklığını artırır ve böylece çevreye karşı daha esnek bir tepki uzayı oluşturur.*

Bu ikinci seçenek, modern bilişsel sistemlerin ve özellikle yapay zekâ mimarilerinin neden giderek daha karmaşık ve çok katmanlı hale geldiğini açıklayan temel ilkedir. Zihin artık tek bir merkezden çalışan basit bir kontrol sistemi olarak düşünülemez; bunun yerine farklı soyutlama seviyelerinde çalışan çoklu alt sistemlerin oluşturduğu bir yapı olarak görülmelidir. Algısal süreçler, temsiller, dikkat mekanizmaları ve karar süreçleri birbirinden ayrışmış ama etkileşim halinde olan modüller olarak çalışır. Bu modülerlik, aslında Ashby'nin "gereken çeşitlilik" ilkesinin bilişsel karşılığıdır.

Karpathy'nin modern metin üretim sistemi çerçevesi bu noktada Ashby'nin fikrini pratik bir düzeye taşır. Belirtme, doğrulama ve çalışma zamanı katmanları aslında sistemin içsel çeşitliliğini organize etme biçimidir. Belirtme, sistemin hangi alanı kapsaması gerektiğini tanımlar; çalışma zamanı, bu kapsama alanında gerçek üretimi gerçekleştirir; doğrulama ise sistemin çevreye verdiği yanıtın yeterli olup olmadığını sürekli kontrol eder. Bu üçlü yapı, Ashby'nin soyut olarak tanımladığı çeşitlilik gerekliliğini operasyonel bir mimariye dönüştürür.

Bu perspektiften bakıldığında Ashby'nin katkısı yalnızca sibernetik teoriyi genişletmek değildir; aynı zamanda düşünmenin neden basit yapıların ötesine geçtiğini açıklamaktır. *Çünkü dünya basit değildir ve basit sistemler karmaşık dünyalar karşısında kaçınılmaz olarak başarısız olur.* Bu nedenle zihin, tarihsel olarak giderek daha fazla katmana bölünmüş, daha modüler ve daha esnek hale gelmiştir. Ashby'nin yasası bu evrimin altında yatan zorunluluğu açıklar: karmaşıklık, ancak eşdeğer karmaşıklıkla yönetilebilir.

Sonuç olarak bu çerçeve bize şunu söyler: düşünme, yalnızca bilgi işleme değildir; aynı zamanda karmaşıklıkla eşleşme problemidir. Ve bu eşleşme, tek bir mekanizma ile değil, çoklu ve dağıtık bir içsel yapı ile mümkün olur.

Bu bakış açısı, düşünmenin doğasını kökten değiştirir. Düşünme artık yalnızca doğru sonuç üretme süreci değildir; aynı zamanda belirsizlikle baş etme, karmaşıklığı azaltma ve çevreyi yönetilebilir hale getirme sürecidir. Zihin, dış dünyayı birebir kopyalamaz; onun yerine, bu

dünyayla etkileşim kurabileceği sadeleştirilmiş modeller üretir. Bu modeller, gerçekliğin kendisi değil, onunla başa çıkmayı mümkün kılan temsillerdir.

Wiener ve Ashby birlikte ele alındığında, zihnin artık kapalı bir hesaplama sistemi olarak değil, açık bir kontrol ve uyum sistemi olarak düşünülmesi gerektiği ortaya çıkar. Bu sistemde her bileşen, çevreyle sürekli etkileşim halindedir ve bu etkileşimler geri besleme döngüleri aracılığıyla sistemin davranışını belirler. Bu nedenle düşünme, doğrusal bir süreç değil, döngüsel bir dinamik haline gelir.

Bu dönüşüm, felsefi düzeyde önemli bir kırılmayı temsil eder. Çünkü artık zihin, açıklanacak bir varlık olmaktan çıkar ve tasarlanabilir bir sistem haline gelir. Eğer bir sistem geri besleme döngüleriyle tanımlanabiliyorsa, bu döngüler farklı şekillerde kurulabilir. Eğer sistemin karmaşıklık ihtiyacı ölçülebiliyorsa, bu ihtiyaç farklı mimarilerle karşılanabilir. *Bu noktada düşünme, yalnızca bir "ne" sorusu olmaktan çıkar ve "nasıl bir sistem olabilir" sorusuna dönüşür.*

Bu dönüşümün önemi, felsefeden mühendisliğe doğru bir kayma yaratmasındadır. Geleneksel felsefi yaklaşım zihni anlamaya çalışırken, sistem yaklaşımı zihni inşa edilebilir bir yapı olarak ele alır. Bu, yalnızca epistemolojik bir değişim değil, aynı zamanda metodolojik bir değişimdir. Artık zihni anlamak için yalnızca gözlem yapmak yeterli değildir; aynı zamanda model kurmak, simülasyon yapmak ve alternatif sistemler tasarlamak gerekir.

Bu noktada "zihin = sistem" fikri, basit bir metafor olmaktan çıkar ve güçlü bir mühendislik sezgisine dönüşür. Zihin artık girişleri alan, işleyen ve çıktı üreten tekil bir yapı değil; sürekli geri besleme alan, çevreyle etkileşen ve kendi içsel yapısını bu etkileşimler üzerinden yeniden düzenleyen çok katmanlı bir organizasyondur. Bu organizasyon içinde her alt süreç, diğer süreçlerle etkileşim halindedir ve bu etkileşimler sistemin genel davranışını belirler.

Bu çerçeve, modern yapay zekâ ve bilişsel mimarilerin temelini oluşturur. Çünkü artık mesele yalnızca akıllı davranış üretmek değil, bu davranışı üreten sistemin nasıl organize edildiğini anlamaktır. Bu organizasyon, belirli kuralların ötesinde, geri besleme, çeşitlilik ve modülerlik gibi ilkeler üzerine kuruludur. Böylece düşünme, içerik üretiminden çok sistem tasarımı problemine dönüşür.

Sistem fikrinin doğuşu bu nedenle yalnızca tarihsel bir aşama değil, düşünmenin doğasına dair temel bir yeniden tanımlamadır. Bu aşamadan sonra zihni anlamak, onu bir nesne olarak incelemekten çok, bir yapı olarak tasarlamayı içerir. Ve bu tasarım fikri, Aristoteles'in mantıksal yapısından Turing'in hesaplanabilir makinelerine, oradan Wiener ve Ashby'nin sistem teorilerine kadar uzanan uzun bir dönüşümün doğal sonucudur.

Kaynaklar

Ashby, W. R. (1956). *An introduction to cybernetics*. Chapman & Hall.

Bertalanffy, L. von. (1968). *General system theory: Foundations, development, applications*. George Braziller.

Beer, S. (1959). *Cybernetics and management*. English Universities Press.

Chomsky, N. (1957). *Syntactic structures*. Mouton.

Descartes, R. (1998). *Discourse on method and meditations on first philosophy* (D. A. Cress, Trans.). Hackett Publishing. (Original works published 1637, 1641)

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63*(2), 81–97. https://doi.org/10.1037/h0043158

Minsky, M. (1967). *Computation: Finite and infinite machines*. Prentice-Hall.

Minsky, M. (1986). *The society of mind*. Simon & Schuster.

Neisser, U. (1967). *Cognitive psychology*. Appleton-Century-Crofts.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Prentice-Hall.

Norbert Wiener. (1948). *Cybernetics: Or control and communication in the animal and the machine*. MIT Press.

Powers, W. T. (1973). *Behavior: The control of perception*. Aldine.

Simon, H. A. (1996). *The sciences of the artificial* (3rd ed.). MIT Press.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, 42*(2), 230–265. https://doi.org/10.1112/plms/s2-42.1.230

Turing, A. M. (1950). Computing machinery and intelligence. *Mind, 59*(236), 433–460. https://doi.org/10.1093/mind/LIX.236.433

Wiener, N. (1950). *The human use of human beings: Cybernetics and society*. Houghton Mifflin.

Varela, F. J., Thompson, E., & Rosch, E. (1991). *The embodied mind: Cognitive science and human experience*. MIT Press.

Ashby, W. R. (1958). Requisite variety and its implications for the control of complex systems. *Cybernetica, 1*, 83–99.

Beer, S. (1972). *Brain of the firm*. Allen Lane.

Holland, J. H. (1992). *Adaptation in natural and artificial systems*. MIT Press.

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society* (pp. 1–12). Lawrence Erlbaum Associates.

Karpathy, A. (2025). *Software 3.0* [Conference presentation]. AI Startup School, San Francisco, CA.

3. BİLİŞSEL DEVRİM

Bilişsel devrim, düşünme tarihindeki en sessiz ama en köklü kırılmalardan biridir. Davranışçılığın uzun süre egemen olduğu bir dönemde, zihin "içeriği bilinmeyen bir kara kutu" olarak kabul edilirken, 20. yüzyılın ortasında bu kara kutunun yeniden açılmasıyla birlikte düşünme, ilk kez sistematik olarak içsel süreçler üzerinden açıklanmaya başlanır. Bu dönüşüm, yalnızca psikolojide bir paradigma değişimi değildir; aynı zamanda zihnin yeniden mühendislik nesnesi olarak ele alınmasının da başlangıcıdır. Artık mesele yalnızca davranışı gözlemek değil, davranışı üreten içsel mimariyi modellemektir.

Bu yeni yaklaşımın merkezinde tek bir fikir vardır: insan zihni, bilgi işleyen bir sistemdir. Ancak bu ifade basit bir metafor değil, giderek daha formel hale gelen bir araştırma programıdır. Zihin artık dışsal uyarıcılara verilen tepkilerin toplamı değil, içsel temsiller, süreçler ve algoritmik yapılar üzerinden çalışan bir sistem olarak düşünülür. Bu noktada bilişsel devrim, sistem fikrinin zihne uygulanmış en doğrudan formudur.

Allen Newell ve Herbert Simon'un çalışmaları bu dönüşümün en somut başlangıç noktasıdır. 1950'lerin sonu ve 1960'ların başında ortaya koydukları çalışmalar, yapay zekâ tarihinin yalnızca ilk büyük başarıları arasında yer almaz; aynı zamanda düşünmenin mühendislik diliyle yeniden tanımlanmasının da başlangıcını oluşturur. O güne kadar zihin çoğunlukla psikologların, filozofların ve nörologların araştırma konusu olmuştu. *Newell ve Simon ise farklı bir soru sordular: Eğer insan problem çözebiliyorsa, bu süreç bilgisayar üzerinde yeniden üretilebilir mi? Daha da önemlisi, problem çözme davranışının altında yatan genel mekanizmalar tanımlanabilir mi?*

Bu soruların sonucunda geliştirdikleri General Problem Solver (GPS – Genel Problem Çözücü) sistemi, insan problem çözme davranışını sembolik bir arama süreci olarak modellemeye çalışan ilk kapsamlı girişimlerden biri oldu. GPS'in temel varsayımı oldukça basit görünüyordu: Bir problem, mevcut durum ile ulaşılmak istenen hedef durum arasındaki farktır. Düşünmenin görevi ise bu farkı giderek azaltacak işlemleri bulmak ve uygulamaktır. Böylece düşünme, mistik veya açıklanamaz bir faaliyet olmaktan çıkarak, belirli kurallarla yönetilen bir arama ve karar verme süreci olarak tanımlanmaya başlandı.

GPS sisteminde zihin, olası durumlar uzayında hareket eden bir gezgin gibidir. Her karar yeni bir duruma götürür, her durum yeni seçenekler üretir ve sistem sürekli olarak hedefe yaklaşıp yaklaşmadığını değerlendirir. Bir satranç oyuncusunun hamleleri hesaplaması, bir matematikçinin ispat kurması veya bir öğrencinin karmaşık bir problemi çözmesi bu çerçevede benzer süreçler olarak görülür. Hepsinde ortak olan şey, mevcut durum ile hedef arasındaki farkı azaltmaya yönelik sistematik bir arama yürütülmesidir.

Bu yaklaşımın önemi yalnızca bir program geliştirilmiş olmasında değildir. Daha derinde, düşünmenin ilk kez açık biçimde bir mühendislik problemi olarak ele alınmasındadır. Çünkü GPS, "İnsan nasıl düşünür?" sorusunu "Bir sistem problemi nasıl çözer?" sorusuna dönüştürür. Böylece zihinsel süreçler ilk kez analiz edilebilir, modellenebilir ve bilgisayar üzerinde simüle

edilebilir hale gelir. Bu, Aristoteles'in mantığından Turing'in hesaplanabilirlik kuramına uzanan çizginin pratik bir mühendislik uygulamasına dönüşmesidir.

Newell ve Simon'un yaklaşımında belki de en kritik dönüşüm şudur: düşünme artık içerik değil, süreçtir. Geleneksel yaklaşımda insanların ne düşündüğü, hangi fikirlere sahip olduğu veya hangi bilgilere ulaştığı önemsenirken, Newell ve Simon için asıl önemli olan zihnin bu sonuçlara hangi işlemler aracılığıyla ulaştığıdır. Başka bir deyişle, sonuçtan çok süreç incelenir. *Bu değişim bugün bize doğal görünse de o dönemde son derece radikaldi. Çünkü ilk kez düşünmenin kendisi bir algoritma olarak ele alınıyordu.*

Bu bakış açısı, modern bilgisayar biliminin ve yapay zekânın temelini oluşturan algoritmik düşünce anlayışını da şekillendirdi. GPS'in kullandığı yöntemler —arama ağaçları, hedef-alt hedef ilişkileri, problem uzayları ve sezgisel stratejiler— bugün hâlâ birçok yapay zekâ sisteminin temel bileşenleri arasında yer alır. Daha da önemlisi, bu yaklaşım insan zihninin karmaşık görünen davranışlarının altında sistematik ve tekrar edilebilir süreçler bulunduğunu ileri sürer.

Newell ve Simon'un geliştirdiği means-ends analysis (amaç-araç analizi) yöntemi bu yaklaşımın merkezindedir. Sistem önce mevcut durum ile hedef durum arasındaki farkı belirler. Daha sonra bu farkı azaltabilecek işlemleri araştırır. Eğer doğrudan bir çözüm yoksa, yeni alt hedefler oluşturur ve bunları çözmeye çalışır. Böylece problem çözme, tek bir büyük sıçrama değil, hiyerarşik olarak örgütlenmiş çok sayıda küçük kararın birleşimi haline gelir. Bu yapı, daha sonra bilişsel psikolojide, uzman sistemlerde ve hatta modern görev planlama algoritmalarında yaygın olarak kullanılacaktır.

Bu noktada ortaya çıkan fikir son derece önemlidir: İnsan zihni karmaşık problemleri doğrudan çözmez; onları daha küçük ve yönetilebilir alt problemlere böler. Bugün bir araştırmacının makale yazarken, bir mühendisin ürün geliştirirken veya bir öğrencinin tez hazırlarken izlediği yöntem büyük ölçüde aynı mantığa dayanır. Karmaşık hedefler küçük hedeflere ayrılır, bu hedefler çözülür ve sonuçlar birleştirilir.

Aslında Karpathy'nin günümüzde önerdiği düşünme sistemleri ile Newell ve Simon'un yaklaşımı arasında dikkat çekici bir süreklilik vardır. GPS'te problem çözme süreci durumlar ve hedefler arasında organize edilirken, Karpathy'nin çerçevesinde düşünme süreci belirtme (specification), doğrulama (verification) ve çalışma zamanı (runtime) katmanları arasında organize edilir. Her iki yaklaşımın ortak noktası, düşünmeyi tek parça bir olay olarak değil, açıkça tanımlanabilen ve yönetilebilen aşamalara ayrılmış bir süreç olarak görmeleridir.

Ancak Karpathy'nin sistemi bir bakıma GPS'in modern bir uzantısıdır. Newell ve Simon, insanın problemi nasıl çözdüğünü anlamaya çalışıyordu. Karpathy ise insan ve yapay zekânın birlikte problem çözmesini sağlayacak üretim mimarileri tasarlamaktadır. GPS'te hedefe ulaşmak için problem uzayında arama yapılırken, Karpathy'nin sisteminde model önce belirtimi yorumlar, ardından çıktı üretir ve daha sonra doğrulama mekanizmalarıyla kendi üretimini değerlendirir.

Bu süreç, GPS'in problem çözme döngüsünün çok daha güçlü ve ölçeklenebilir bir versiyonu gibi görülebilir.

Newell ve Simon'un en büyük iddiası ise bugün bile tartışılmaya devam etmektedir: İnsan düşünmesi sembolik manipülasyonlara indirgenebilir. Onlara göre zihin özünde semboller üzerinde işlem yapan fiziksel bir sistemdir. *Daha sonra geliştirdikleri ünlü Physical Symbol System Hypothesis (Fiziksel Sembol Sistemi Hipotezi) bu görüşü açık biçimde ifade eder. Hipoteze göre, genel zekâ üretebilen her sistem semboller üzerinde işlem yapmak zorundadır ve sembolleri işleyebilen her sistem prensipte genel zekâ geliştirebilir.*

Bu iddia, yapay zekâ tarihindeki en etkili fikirlerden biri haline gelir. Çünkü burada zihin ilk kez açık biçimde bir bilgi işleme sistemi olarak tanımlanmaktadır. Her ne kadar daha sonra bağlantısalcı yaklaşımlar ve derin öğrenme hareketi bu görüşe ciddi eleştiriler getirecek olsa da düşünmenin mühendislik nesnesi olarak ele alınması fikri büyük ölçüde Newell ve Simon'un çalışmalarıyla kurumsallaşmıştır.

Sonuç olarak Newell ve Simon, düşünmeyi açıklamaktan çok modellemeye çalışan ilk büyük araştırmacılar olarak görülebilir. Onların çalışmalarıyla birlikte düşünme, felsefenin soyut tartışmalarından çıkarak bilgisayar programlarına, algoritmalara ve bilişsel mimarilere taşınmıştır. Eğer Turing bize düşünmenin hesaplanabilir olabileceğini göstermişse, Newell ve Simon düşünmenin nasıl sistematik olarak modellenebileceğini göstermiştir. Bu nedenle modern düşünme mimarilerinin, bilişsel sistemlerin ve hatta Karpathy'nin önerdiği yeni nesil üretim sistemlerinin köklerinde Newell ve Simon'un açtığı yol bulunmaktadır.

Herbert Simon bu çerçeveyi daha da derinleştirerek “sınırlı rasyonalite” kavramını geliştirir. Klasik ekonomi ve rasyonalite modelleri, insanın tüm seçenekleri değerlendirebilen ideal bir optimizasyon makinesi olduğunu varsayar. Simon ise bunun gerçekçi olmadığını gösterir. İnsan zihni sınırlı bellek, sınırlı dikkat ve sınırlı zaman içinde çalışır. Bu nedenle optimum çözümler üretmek yerine, yeterince iyi olan çözümlerle yetinir. Bu yaklaşım, zihnin temel doğasını değiştirir: zihin artık mükemmel çözüm arayan bir sistem değil, kısıtlar altında çalışan bir karar mekanizmasıdır.

Bu fikir, düşünme mimarisi açısından son derece kritiktir. Çünkü burada zihin ilk kez “kusurlu ama işlevsel” bir sistem olarak tanımlanır. Bu kusur, bir hata değil, yapısal bir özelliktir. Zihin, sınırlı kaynaklar içinde çalıştığı için, çözüm uzayını daraltmak zorundadır. Bu da bilişsel süreçlerin neden heuristikler, sezgiler ve yaklaşık yöntemler içerdiğini açıklar. Simon’un katkısı, zihni ideal bir hesaplama sistemi olmaktan çıkarıp gerçek bir mühendislik sistemi haline getirmesidir.

Bu dönüşümün psikoloji alanındaki yansıması Ulric Neisser ile birlikte ortaya çıkar. Neisser, bilişsel psikolojiyi bağımsız bir bilim dalı olarak kurarken, zihnin içsel süreçlerinin bilimsel olarak incelenebilir olduğunu savunur. Davranışçılığın aksine, artık yalnızca dışsal davranışlar değil, bu davranışları üreten zihinsel temsiller ve süreçler de araştırma nesnesi haline gelir. Zihin, yeniden içsel bir mimari olarak tanımlanır.

Neisser'in yaklaşımı, bilişsel devrimin epistemolojik temelini oluşturur. Çünkü burada zihin, gözlemlenemeyen bir kara kutu olmaktan çıkar ve teorik modellerle açıklanabilir bir sistem haline gelir. Bu modeller, doğrudan gözlemlenemese bile davranış üzerinden çıkarım yapılabilen içsel yapıları temsil eder. Böylece zihinsel süreçler, mühendislik sistemleri gibi modellenebilir hale gelir.

George Miller'ın çalışmaları, bilişsel devrimin en önemli dönüm noktalarından birini oluşturur çünkü Miller, zihnin yalnızca nasıl çalıştığını değil, aynı zamanda hangi sınırlar içinde çalıştığını araştırmıştır. 1956 yılında yayımladığı ünlü makalesi *"The Magical Number Seven, Plus or Minus Two" (Sihirli Sayı Yedi, Artı Eksi İki)*, insan çalışma belleğinin kapasitesine ilişkin ilk sistematik bulgulardan birini ortaya koyar. *Miller, insanların aynı anda akıllarında tutabildikleri bilgi miktarının sınırsız olmadığını göstermiştir. Deneyler, insanların çoğu durumda yaklaşık beş ila dokuz bilgi birimini aynı anda işleyebildiğini ortaya koyuyordu. Bu bilgi birimleri tek tek rakamlar, harfler veya sözcükler olabileceği gibi, daha büyük anlam kümeleri de olabilirdi.*

Bu bulgu ilk bakışta basit bir psikoloji sonucu gibi görünebilir. Ancak etkisi son derece derindir. Çünkü o güne kadar insan zihni çoğu zaman soyut ve sınırsız bir düşünme alanı olarak tasavvur ediliyordu. Miller ise zihnin çok somut kapasite sınırlarına sahip olduğunu gösterdi. İnsanlar aynı anda yalnızca belirli miktarda bilgiyi aktif olarak işleyebiliyor, daha fazlası ise dikkat kaybına, unutmaya veya hata yapmaya yol açıyordu. Böylece zihnin yalnızca ne yaptığı değil, neyi yapamayacağı da bilimsel olarak görünür hale geldi.

Miller'ın çalışmalarının bir başka önemli sonucu da "chunking" olarak bilinen kümeleme kavramının ortaya çıkmasıdır. İnsanlar yedi ayrı rakamı akılda tutmakta zorlanabilirken, bu rakamlar anlamlı gruplar halinde düzenlendiğinde çok daha büyük bilgi miktarlarını yönetebilirler. *Örneğin 1-9-2-3-1-9-4-5 gibi sekiz rakamı tek tek hatırlamak zor olabilir; ancak bunları 1923 ve 1945 şeklinde iki tarih olarak kodlamak çok daha kolaydır. Bu durum bize zihnin yalnızca depolama yapan pasif bir sistem olmadığını, bilgiyi yeniden organize ederek kapasitesini etkin biçimde kullandığını gösterir. Başka bir deyişle, zekânın önemli bir kısmı daha fazla bilgiye sahip olmak değil, bilgiyi daha iyi yapılandırabilmektir.*

Miller'ın bulguları, düşünme mimarisi açısından son derece kritik bir sonuca işaret eder: Her sistem gibi zihnin de darboğazları vardır. Nasıl bir fabrikanın üretim kapasitesi makinelerinin hızına bağlıysa, nasıl bir bilgisayarın performansı işlemci ve bellek sınırlarıyla belirleniyorsa, insan düşünmesi de çalışma belleğinin kapasitesi tarafından şekillendirilir. Bir problemi çözerken, bir makale yazarken veya karmaşık bir tartışmayı takip ederken karşılaştığımız zorlukların önemli bir bölümü aslında bu kapasite sınırlarından kaynaklanır. İnsan zihni sonsuz bilgi işleyen bir yapı değil, dikkat ve bellek kaynaklarını dikkatle kullanmak zorunda olan sınırlı bir sistemdir.

Miller'ın katkısı tam da burada ortaya çıkar. O, zihni yalnızca işleyen bir sistem olarak değil, aynı zamanda kapasitesi sınırlı bir mimari olarak tanımlar. Bu, bilişsel bilimler açısından olduğu kadar mühendislik açısından da son derece önemli bir bakış açısıdır. Çünkü mühendislikte herhangi bir sistem tasarlanırken ilk sorulardan biri kapasite sorusudur: Ne kadar veri

işlenebilir? Ne kadar bellek kullanılabilir? Aynı anda kaç işlem yürütülebilir? Miller'ın çalışmaları, benzer soruların insan zihni için de geçerli olduğunu göstermiştir.

Bu nedenle Miller'dan sonra bellek, dikkat ve işlem yükü gibi kavramlar yalnızca psikolojik olgular olmaktan çıkıp mühendislik parametreleri olarak da görülmeye başlanmıştır. İnsan zihni artık gizemli bir "kara kutu" değil, kaynakları sınırlı bir bilgi işleme sistemi olarak ele alınabilir hale gelmiştir. Bu yaklaşım daha sonra bilişsel mimarilerin, insan-bilgisayar etkileşiminin ve hatta eğitim tasarımının temelini oluşturacaktır. Günümüzde kullanıcı arayüzlerinin sade tasarlanması, ders içeriklerinin küçük parçalara bölünmesi veya karmaşık görevlerin alt adımlara ayrılması gibi uygulamaların arkasında büyük ölçüde Miller'ın ortaya koyduğu bu kapasite gerçeği bulunmaktadır.

Karpathy'nin düşünme sistemleri açısından bakıldığında ise Miller'ın önemi daha da belirginleşir. Karpathy'nin önerdiği belirtme (specification), doğrulama (verification) ve çalışma zamanı (runtime) ayrımı aslında insan zihninin bu kapasite sınırlarını aşmaya yönelik bir organizasyon biçimi olarak da okunabilir. Geleneksel çalışmada insan aynı anda problemi anlamaya, çözmeye, yazmaya ve kontrol etmeye çalışır. Bu durum çalışma belleği üzerinde büyük bir yük oluşturur. *Karpathy'nin yaklaşımı ise bu görevleri katmanlara ayırır. Önce ne yapılacağı tanımlanır, sonra üretim gerçekleştirilir, ardından doğrulama yapılır. Böylece tek bir zihinsel süreç içinde birbirine karışan görevler ayrıştırılır ve bilişsel yük azaltılır.*

Bu açıdan Miller'ın çalışmaları yalnızca insan belleğinin sınırlarını açıklamaz; aynı zamanda neden düşünme süreçlerinin yapılandırılması gerektiğini de açıklar. Çünkü karmaşık problemlerin çözümü çoğu zaman daha zeki olmaktan değil, sınırlı bilişsel kaynakları daha iyi organize etmekten geçer. Karpathy'nin modern üretim sistemleri ile Miller'ın bilişsel kapasite anlayışı burada birleşir: İster insan zihni olsun ister yapay zekâ sistemi, başarılı düşünme yalnızca hesaplama gücüne değil, bu gücün nasıl organize edildiğine bağlıdır.

Sonuç olarak Miller'ın ortaya koyduğu temel gerçek şudur: Zihin sonsuz bir düşünme alanı değildir. Belirli kapasitelere, darboğazlara ve kaynak sınırlarına sahip bir sistemdir. Düşünme mimarisini anlamak isteyen herkes için bu gerçek kritik öneme sahiptir. Çünkü bir sistemi tasarlamadan önce onun sınırlarını bilmek gerekir. Miller, bilişsel bilime tam da bu sınırları göstermiş ve düşünmenin mühendislik olarak yeniden tanımlanmasında vazgeçilmez bir halka oluşturmuştur.

Noam Chomsky'nin katkısı ise bilişsel devrimi yalnızca süreçler ve problem çözme mekanizmaları düzeyinde bırakmaz; tartışmayı çok daha derin bir yapısal seviyeye taşır. Newell, Simon ve Miller gibi araştırmacılar zihnin nasıl bilgi işlediğini ve problemleri nasıl çözdüğünü açıklamaya çalışırken, Chomsky çok daha temel bir soru sorar: İnsan zihni neden dili bu kadar hızlı ve başarılı biçimde öğrenebilmektedir? Bir çocuk, son derece karmaşık bir dil sistemini birkaç yıl içinde nasıl edinmektedir? Üstelik bu öğrenme süreci eksik, hatalı ve düzensiz girdilere rağmen nasıl gerçekleşmektedir?

1957 yılında yayımlanan *Syntactic Structures (Sözdizimsel Yapılar)* ve daha sonra gelen *Aspects of the Theory of Syntax (Sözdizimi Kuramının Görünümleri)* gibi eserlerinde Chomsky, o dönemde egemen olan davranışçı açıklamalara güçlü bir meydan okuma getirir. Davranışçılara göre dil, çevreden alınan uyarıcıların ve pekiştirmelerin sonucu olarak öğrenilen bir davranıştı. Chomsky ise bunun yeterli bir açıklama olmadığını savunur. Çocukların karşılaştıkları dil girdisi, dilin bütün karmaşıklığını öğrenmelerine yetecek kadar zengin değildir. Buna rağmen çocuklar daha önce hiç duymadıkları cümleleri kurabilir, dilin kurallarını yaratıcı biçimde kullanabilir ve sonsuz sayıda yeni ifade üretebilirler.

Bu gözlem Chomsky'yi son derece radikal bir sonuca götürür: Dil yalnızca öğrenilen bir davranış değildir; insan zihni dil öğrenmeye uygun belirli bir yapısal mimariyle dünyaya gelir. İşte onun ünlü Evrensel Gramer (Universal Grammar) teorisi bu noktada ortaya çıkar. *Chomsky'ye göre insan beyninde, bütün doğal dillerin temelini oluşturan ortak yapısal ilkeler bulunmaktadır. Çocuk dil öğrenirken sıfırdan bir sistem kurmaz; zaten mevcut olan bilişsel bir mimariyi çevresel verilerle doldurur ve ayarlar.*

Bu düşünce bilişsel devrim içinde son derece güçlü bir yapısal iddiadır. Çünkü burada zihin yalnızca bilgi işleyen bir süreçler bütünü olarak değil, aynı zamanda önceden tanımlanmış bir mimari olarak görülmektedir. Newell ve Simon zihnin nasıl çalıştığını anlamaya çalışırken, Chomsky zihnin hangi temel tasarımla çalıştığını araştırır. Böylece bilişsel bilimlerde önemli bir vurgu değişimi yaşanır: Sorun artık yalnızca işlem süreçlerini açıklamak değildir; bu süreçlerin üzerinde çalıştığı yapısal altyapıyı da anlamaktır.

Chomsky'nin yaklaşımında dil, yüzeyde görülen sözcük dizilerinden çok daha derin bir organizasyona sahiptir. Ona göre insanlar konuşurken yalnızca kelimeleri yan yana dizmezler; görünmeyen kurallar, hiyerarşik ilişkiler ve soyut yapılar üzerinde işlem yaparlar. Bu nedenle dil üretimi, basit bir ezberleme veya çağrışım süreci değildir. İnsan zihni, görünmeyen yapıları kullanarak sonsuz sayıda yeni cümle oluşturabilir. Chomsky bunu açıklamak için yüzey yapı (surface structure) ve derin yapı (deep structure) kavramlarını geliştirir. Her ne kadar bu kavramlar daha sonra farklı biçimlerde yeniden yorumlanmış olsa da, temel fikir etkisini korur: Zihnin görünür davranışlarının altında daha derin organizasyon katmanları bulunmaktadır.

Bu görüş, düşünme mimarisi açısından son derece önemlidir. Çünkü ilk kez zihnin yalnızca deneyimlerden oluşmadığı, aynı zamanda belirli ön koşullarla dünyaya geldiği ileri sürülmektedir. Başka bir ifadeyle, düşünme tamamen çevrenin ürünü değildir. Öğrenme elbette önemlidir, ancak öğrenmenin gerçekleşebilmesi için önceden var olan bir mimarinin bulunması gerekir. Bu durum günümüzde makine öğrenmesi ve yapay zekâ alanında da karşımıza çıkan bir soruya benzer: Bir sistemin öğrenebilmesi için önceden hangi yapısal özelliklere sahip olması gerekir?

Chomsky'nin çalışmaları, düşünmenin yalnızca öğrenilen bir süreç olmadığını, aynı zamanda önceden belirlenmiş yapısal kısıtlar üzerine kurulduğunu göstermeye çalışır. İnsan zihni boş bir levha değildir; belirli bilişsel eğilimlerle, belirli organizasyon ilkeleriyle ve belirli işlem

sınırlarıyla donatılmıştır. Dil üretimini mümkün kılan şey yalnızca sözcük dağarcığı değil, bu sözcükleri organize eden görünmez yapısal mekanizmalardır.

Bu yaklaşım zihni daha da mühendislik nesnesine yaklaştırır. Çünkü mühendislikte herhangi bir sistemi anlamanın iki yolu vardır: Sistemin çalışma süreçlerini incelemek ve sistemin mimarisini incelemek. Chomsky ikinci yolu öne çıkarır. Artık soru yalnızca "zihin nasıl çalışıyor?" değildir. Aynı zamanda "zihin hangi yapısal tasarım ilkeleri üzerine kurulmuştur?" sorusu da önem kazanır. Böylece bilişsel bilimlerde süreç odaklı açıklamalar ile mimari odaklı açıklamalar birleşmeye başlar.

Karpathy'nin düşünme sistemleri bağlamında bakıldığında da Chomsky'nin etkisi dolaylı olarak hissedilir. *Karpathy, büyük dil modellerinin ortaya çıkardığı yeni düşünme mimarilerini tartışırken sürekli olarak yapı ile süreç arasındaki ilişkiye dikkat çeker.* Bir modelin başarısı yalnızca eğitim verisinden kaynaklanmaz; modelin mimarisi de belirleyici rol oynar. Transformer mimarilerinin başarısı, veriyi işleme biçimlerinden olduğu kadar, veriyi organize etme biçimlerinden de kaynaklanır. *Bu açıdan bakıldığında Chomsky'nin temel sorusu hâlâ günceldir: Zekâyı mümkün kılan şey yalnızca öğrenme midir, yoksa öğrenmeyi mümkün kılan belirli yapısal düzenlemeler de var mıdır?*

Sonuç olarak Chomsky, bilişsel devrime yeni bir boyut kazandırmıştır. Newell ve Simon düşünmeyi bir süreç olarak, Miller kapasiteyle sınırlı bir sistem olarak açıklarken, Chomsky zihni belirli yapısal ilkeler üzerine kurulmuş bir mimari olarak ele almıştır. Böylece düşünme ilk kez üç boyutlu bir çerçevede görülmeye başlanır: süreçler, kapasite sınırları ve yapısal organizasyon. Modern düşünme mimarileri, yapay zekâ sistemleri ve büyük dil modelleri üzerine yapılan tartışmaların önemli bir kısmı hâlâ bu üç eksen etrafında dönmektedir. Chomsky'nin kalıcı etkisi de tam burada yatar: O, düşünmenin yalnızca nasıl gerçekleştiğini değil, hangi görünmez mimari sayesinde mümkün hale geldiğini sorgulamıştır.

Bilişsel devrim bu haliyle tek bir sonucu açıkça ortaya koyar: zihin, içsel olarak modellenebilir bir sistemdir. Bu sistem, sınırlı kaynaklarla çalışır, sembolik temsiller kullanır, problem çözme stratejileri uygular ve belirli yapısal kısıtlar içinde faaliyet gösterir. Dolayısıyla düşünme artık yalnızca felsefi bir konu değil, mühendislik ve bilimsel modelleme alanına giren bir problem haline gelir.

Bu bölümün genel sonucu şudur: bilişsel devrim, zihni dışsal davranışlardan içsel mimariye taşır. Bu taşınma, düşünme kavramını radikal biçimde değiştirir. Artık zihin, yalnızca ne yaptığıyla değil, nasıl bir yapı olduğu ile tanımlanır. Ve bu yapı, ilerleyen bölümlerde sistem teorisi ve yapay zekâ ile birleşerek daha da karmaşık bir üretim mimarisine dönüşecektir.

Kaynaklar

Chomsky, N. (1957). *Syntactic structures*. Mouton.

Chomsky, N. (1965). *Aspects of the theory of syntax*. MIT Press.

Chomsky, N. (1968). *Language and mind*. Harcourt, Brace & World.

Fodor, J. A. (1975). *The language of thought*. Harvard University Press.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63*(2), 81–97. https://doi.org/10.1037/h0043158

Miller, G. A., Galanter, E., & Pribram, K. H. (1960). *Plans and the structure of behavior*. Holt, Rinehart and Winston.

Minsky, M. (1967). *Computation: Finite and infinite machines*. Prentice-Hall.

Minsky, M. (1986). *The society of mind*. Simon & Schuster.

Newell, A., & Simon, H. A. (1959). GPS, a program that simulates human thought. In H. Billing (Ed.), *Proceedings of the International Conference on Information Processing* (pp. 216–223). UNESCO.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Prentice-Hall.

Neisser, U. (1967). *Cognitive psychology*. Appleton-Century-Crofts.

Nilsson, N. J. (1998). *Artificial intelligence: A new synthesis*. Morgan Kaufmann.

Pólya, G. (1945). *How to solve it*. Princeton University Press.

Pylyshyn, Z. W. (1984). *Computation and cognition: Toward a foundation for cognitive science*. MIT Press.

Rumelhart, D. E., McClelland, J. L., & PDP Research Group. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition* (Vols. 1–2). MIT Press.

Simon, H. A. (1955). A behavioral model of rational choice. *Quarterly Journal of Economics, 69*(1), 99–118. https://doi.org/10.2307/1884852

Simon, H. A. (1957). *Models of man: Social and rational*. Wiley.

Simon, H. A. (1972). Theories of bounded rationality. In C. B. McGuire & R. Radner (Eds.), *Decision and organization* (pp. 161–176). North-Holland.

Simon, H. A. (1996). *The sciences of the artificial* (3rd ed.). MIT Press.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind, 59*(236), 433–460. https://doi.org/10.1093/mind/LIX.236.433

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*, 5998–6008.

Wiener, N. (1948). *Cybernetics: Or control and communication in the animal and the machine*. MIT Press.

4. SEMBOLİK ZEKÂ VE KLASİK YZ

Bilişsel devrimin zihni içsel süreçler üzerinden yeniden kurması, kısa süre içinde yapay zekâ alanında çok daha iddialı bir projeye dönüşür: insan düşünmesini açık kurallar ve sembolik temsiller üzerinden yeniden inşa etmek. Bu yaklaşım, *"sembolik yapay zekâ"* olarak adlandırılan dönemin temelini oluşturur. Buradaki ana varsayım oldukça nettir: eğer insan zihni bilgi işleyen bir sistemse, bu sistemin işleyişi semboller, kurallar ve mantıksal dönüşümler üzerinden modellenebilir. Böylece düşünme, biyolojik bir süreç olmaktan çıkar ve formel bir yazılım problemi haline gelir.

Sembolik YZ'nin yükselişi, bilgisayar biliminin erken dönem iyimserliğinin de bir yansımasıdır. Bu dönemde zihin, tıpkı bir program gibi ele alınır; doğru kurallar yazıldığında, doğru davranışların otomatik olarak ortaya çıkacağı varsayılır. Bu bakış açısı, özellikle 1950'ler ve 1960'larda güçlü bir araştırma programına dönüşür. Amaç, insan zekâsını doğrudan kopyalamak değil, onu kurallara dayalı bir sistem olarak yeniden inşa etmektir. Bu sistemde her bilgi bir sembol, her işlem bir kural uygulaması ve her düşünme süreci bir sembolik dönüşüm zinciridir.

Bu yaklaşımın en önemli özelliği, düşünmeyi açık ve izlenebilir hale getirmesidir. Sembolik sistemlerde her adım şeffaftır; sistemin neden belirli bir sonuca ulaştığı açık kurallarla açıklanabilir. Bu durum, erken AI araştırmalarında büyük bir avantaj olarak görülür. Çünkü zihin artık kara kutu olmaktan çıkar, adım adım analiz edilebilir bir yapı haline gelir. Bu şeffaflık, aynı zamanda mühendislik açısından güçlü bir cazibe yaratır: eğer düşünme kurallara indirgenebilirse, o zaman bu kurallar optimize edilebilir, genişletilebilir ve yeniden yazılabilir.

Ancak bu yaklaşımın merkezinde önemli bir varsayım vardır: dünya, sembollerle yeterince iyi temsil edilebilir. Sembolik AI, gerçekliğin karmaşıklığını belirli kategoriler ve kurallar aracılığıyla modellenebileceğini kabul eder. Bu modelleme başarılı olduğu ölçüde sistemler mantıklı davranışlar sergiler, ancak karmaşıklık arttıkça bu yaklaşımın sınırları da görünür hale gelir. Yine de bu dönem, düşünmenin mühendislik tarihindeki en önemli aşamalardan biridir; çünkü zihin ilk kez açıkça "programlanabilir bir sistem" olarak ele alınır.

Marvin Minsky'nin katkıları, sembolik yapay zekâ dönemini yalnızca teknik olarak değil, düşünsel olarak da şekillendiren en önemli unsurlardan biridir. Minsky, bu alanda sadece çalışan sistemler geliştirmekle kalmaz; aynı zamanda "zekâ nasıl organize edilmelidir?" sorusuna da güçlü teorik yanıtlar verir. Onun çalışmaları, özellikle sembolik yapay zekânın daha karmaşık ve insan benzeri hale gelmesi için yeni yollar açar. Bu bağlamda en önemli katkılarından biri "frame theory" (çerçeve kuramı) dır.

Çerçeve kuramının temel fikri oldukça basittir ama etkisi büyüktür: İnsan zihni dünyayı tek tek ve kopuk sembollerle değil, belirli durumlara ait "paketlenmiş bilgi yapılarıyla" anlar. Bir başka deyişle, insanlar bir olayı değerlendirirken tüm bilgiyi sıfırdan kurmazlar; daha önce deneyimledikleri benzer durumları hatırlarlar ve onları yeniden kullanırlar. Minsky'ye göre bu "paketler" ya da "çerçeveler", belirli bir duruma ait beklenen bilgileri içinde taşır. *Örneğin*

“restoran” denildiğinde aklımıza sadece bir kelime gelmez. Aynı anda birçok bilgi otomatik olarak aktive olur: masalar, garsonlar, menü, sipariş verme süreci, yemek yeme, ödeme yapma gibi aşamalar zihnimizde canlanır.

Bu çerçeve yaklaşımı, sembolik yapay zekâ için önemli bir genişlemedir. Çünkü klasik sembolik sistemlerde bilgi genellikle tek tek ve bağımsız semboller şeklinde tutulur. Her bilgi parçası ayrı bir birim gibidir ve sistem bu birimleri kurallar aracılığıyla birleştirir. Ancak çerçeve kuramı, bilginin böyle parçalı değil, bağlam içinde organize edilmiş bütünler halinde temsil edildiğini söyler. Yani bilgi sadece “ne olduğu” değil, aynı zamanda “hangi durumda nasıl çalıştığı” ile birlikte anlam kazanır.

Bu yaklaşım düşünme modelini de daha esnek hale getirir. İnsan zihni dünyayı sürekli olarak tek tek nesneler şeklinde değil, durumlar ve bağlamlar şeklinde işler. Bir restorana girdiğimizde her şeyi yeniden öğrenmeyiz; daha önceki deneyimlerimize dayanarak o ortamı hızlıca anlamlandırırız. Yeni bilgiler geldiğinde de bu çerçeveler güncellenir. Örneğin lüks bir restorana girdiğimizde beklentilerimiz değişir, davranışlarımız farklılaşır. Bu da gösterir ki çerçeveler sabit değildir; yeni deneyimlerle sürekli güncellenen dinamik yapılardır.

Bu bakış açısı, sembolik yapay zekâyı daha “insana benzer” hale getirme yönünde önemli bir adımdır. Çünkü artık sistemler yalnızca mantıksal kurallarla değil, bağlam duyarlı bilgi yapılarıyla çalışmaya başlar. Yani sistemin ne yapacağı sadece “hangi kuralın geçerli olduğuna” değil, “hangi çerçevenin aktif olduğuna” bağlıdır. Bu da düşünmenin daha gerçekçi bir modelini oluşturur.

Minsky’nin en radikal ve en etkili fikri ise “Society of Mind” (Zihin Toplumu) teorisidir. Bu teoriye göre zihin, tek bir merkezden kontrol edilen bütüncül bir yapı değildir. Aksine, çok sayıda küçük, basit ve uzmanlaşmış süreçten oluşur. Her bir süreç ya da “ajan”, zihnin belirli bir görevini yerine getirir. Bazı ajanlar görme ve algıdan sorumludur, bazıları hafızayı yönetir, bazıları karar verme süreçlerinde çalışır, bazıları ise daha düşük seviyeli otomatik işlemleri yürütür.

Bu ajanların hiçbiri tek başına “zekâ” değildir. Zekâ, bu ajanların bir araya gelerek sürekli etkileşim kurmasından ortaya çıkar. Yani zihinde merkezi bir kontrol odası yoktur; bunun yerine birbirleriyle konuşan, iş birliği yapan ve zaman zaman çatışan küçük süreçler vardır. Bu durum bir anlamda insan zihnini bir “toplum” gibi düşünmemizi sağlar. Nasıl bir toplumda farklı bireyler farklı görevler üstleniyorsa, zihinde de farklı ajanlar farklı görevleri yerine getirir.

Bu yaklaşım, düşünme anlayışını kökten değiştirir. Zihin artık tek bir “ben” tarafından yönetilen bir yapı değil, içsel olarak dağıtılmış bir sistemdir. Kararlar, bu ajanların etkileşimi sonucunda ortaya çıkar. Bazen bir ajan baskın olur, bazen birden fazlası birlikte çalışır. Bu nedenle zihinsel süreçler aslında sürekli bir müzakere ve koordinasyon sürecidir.

Minsky’nin Çerçeve Kuramı ve Zihin Toplumu yaklaşımları birlikte düşünüldüğünde, ortaya çok güçlü bir tablo çıkar: Zihin hem bağlamsal çerçeveler içinde çalışan bir bilgi sistemi, hem de çok sayıda küçük sürecin etkileşiminden doğan bir ağdır. Bu iki fikir birlikte, modern yapay zekâ

sistemlerinin de temelini oluşturan çok katmanlı ve dağıtık mimari anlayışına güçlü bir zemin hazırlar.

Bu yaklaşım, zihnin doğasına dair köklü bir değişim önerir. Klasik modellerde zihin tek bir bütünsel sistem olarak ele alınırken, Minsky'nin modelinde zihin parçalı, dağıtık ve rekabetçi bir yapıdır. Bu yapı içinde hiçbir süreç tek başına "zihin" değildir; zihin, bu süreçlerin etkileşiminden doğan bir fenomendir. Bu nedenle düşünme, merkezi bir kontrol mekanizmasının ürünü değil, çoklu süreçlerin dinamik bir dengesi olarak ortaya çıkar.

Zihin Toplumu yaklaşımı, modern yapay zekâ mimarilerinin birçok yönünü önceden sezmiştir. Bugün kullanılan çok ajanlı sistemler, modüler ağ yapıları ve dağıtık hesaplama modelleri, bu fikrin farklı teknik yorumları olarak görülebilir. Minsky'nin temel sezgisi şudur: karmaşık zekâ, basit bileşenlerin doğru organizasyonundan doğar. Bu organizasyon, sabit bir yapı değil, sürekli etkileşim halinde olan bir sistemdir.

Sembolik YZ ile Minsky'nin yaklaşımı birlikte ele alındığında, klasik yapay zekâ paradigmasının temel çerçevesi ortaya çıkar. Bir yanda kurallara dayalı sembolik işlem sistemleri, diğer yanda bu sistemleri bağlam ve modülerlik üzerinden genişleten bir mimari anlayış vardır. Bu iki yaklaşım birlikte, düşünmenin ilk sistematik mühendislik modelini oluşturur.

Ancak bu model, zamanla kendi sınırlarına ulaşacaktır. Çünkü sembolik sistemler, iyi tanımlanmış problemler üzerinde başarılı olurken, gerçek dünyanın belirsiz ve sürekli değişen yapısı karşısında zorlanır. Çerçeve sistemleri bağlamı yakalamaya çalışsa da, dünya sürekli yeni bağlamlar üretir. Zihin Toplumu yapısı esneklik sunsa da, sembolik temsilin katılığı birçok durumda yetersiz kalır.

Yine de bu dönem, düşünme mimarisinin tarihinde kritik bir eşiktir. Çünkü burada ilk kez zihin, açıkça "tasarlanabilir bir sistem" olarak ele alınır. Kurallar yazılabilir, bağlamlar tanımlanabilir ve ajanlar organize edilebilir. Bu, düşünmenin mühendislik nesnesi haline gelmesinin ilk büyük kurumsal ifadesidir.

Sembolik zekâ dönemi bu haliyle yalnızca bir teknolojik aşama değil, aynı zamanda bir düşünme tarzıdır. Bu tarz, zihni açık, modüler ve açıklanabilir bir sistem olarak görür. Ve her ne kadar daha sonra bağlantısal ve istatistiksel yaklaşımlar bu paradigmayı zayıflatacak olsa da sembolik YZ'nin bıraktığı miras kalıcıdır: düşünme, tasarlanabilir bir yapı olarak anlaşılabilir.

Kaynaklar

Bobrow, D. G., & Winograd, T. (1977). An overview of KRL, a knowledge representation language. *Cognitive Science, 1*(1), 3–46.

Feigenbaum, E. A., & Feldman, J. (Eds.). (1963). *Computers and thought*. McGraw-Hill.

Fodor, J. A. (1975). *The language of thought*. Harvard University Press.

Haugeland, J. (1985). *Artificial intelligence: The very idea*. MIT Press.

Minsky, M. (1967). *Computation: Finite and infinite machines*. Prentice-Hall.

Minsky, M. (1974). A framework for representing knowledge. *MIT Artificial Intelligence Laboratory Memo 306*. Massachusetts Institute of Technology.

Minsky, M. (1986). *The society of mind*. Simon & Schuster.

Newell, A., & Simon, H. A. (1959). GPS, a program that simulates human thought. In H. Billing (Ed.), *Proceedings of the International Conference on Information Processing* (pp. 216–223). UNESCO.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Prentice-Hall.

Nilsson, N. J. (1998). *Artificial intelligence: A new synthesis*. Morgan Kaufmann.

Pólya, G. (1945). *How to solve it*. Princeton University Press.

Quillian, M. R. (1968). Semantic memory. In M. Minsky (Ed.), *Semantic information processing* (pp. 227–270). MIT Press.

Rosenblatt, F. (1962). *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Spartan Books.

Russell, S., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals, and understanding*. Lawrence Erlbaum Associates.

Simon, H. A. (1957). *Models of man: Social and rational*. Wiley.

Simon, H. A. (1996). *The sciences of the artificial* (3rd ed.). MIT Press.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind, 59*(236), 433–460. https://doi.org/10.1093/mind/LIX.236.433

Weizenbaum, J. (1976). *Computer power and human reason*. W. H. Freeman.

Winograd, T. (1972). *Understanding natural language*. Academic Press.

5. KRİZ: SEMBOLİK ZİHNİN SINIRLARI

Sembolik yapay zekâ dönemi, zihni açık kurallar, semboller ve bağlamsal çerçeveler üzerinden modelleme konusunda güçlü bir entelektüel program sunmuş olsa da zaman içinde bu yaklaşımın temel bir sınırla karşı karşıya olduğu giderek daha görünür hale gelir. *Bu sınır, teorik bir eksiklikten ziyade, dünyanın kendisinin karmaşıklığından kaynaklanır. Gerçek dünya, önceden tanımlanmış kategorilerle tam olarak kapsanamayacak kadar değişken, belirsiz ve bağlam bağımlıdır. Bu nedenle sembolik sistemlerin en güçlü olduğu yer ile dünyanın en yoğun olduğu yer arasında yapısal bir uyumsuzluk ortaya çıkar.*

Sembolik yapay zekânın temel fikri oldukça basitti ve ilk bakışta son derece mantıklı görünüyordu. Eğer insanlar dünyayı kurallar kullanarak anlıyor ve karar veriyorsa, o zaman bir bilgisayar da aynı şeyi yapabilirdi. Bunun için yapılması gereken şey, dünyaya ilişkin bilgileri semboller halinde tanımlamak ve bu semboller arasında geçerli olan kuralları sisteme öğretmekti. Örneğin bir uzman sistemde "ateş varsa ve öksürük varsa grip olabilir" gibi kurallar yazılabiliyordu. Bilgisayar bu kuralları takip ederek insan uzmanlara benzer kararlar verebiliyordu. 1970'lerde ve 1980'lerde bu yaklaşım öylesine başarılı oldu ki, birçok araştırmacı insan zekâsının büyük ölçüde çözüldüğünü düşünmeye başladı.

Ancak gerçek dünya laboratuvar ortamından çok daha karmaşıktı. Çünkü hayat, birkaç yüz ya da birkaç bin kuralla açıklanabilecek kadar düzenli değildi. Her yeni durum yeni kurallar gerektiriyordu. Her yeni istisna, sisteme ek açıklamalar yapılmasını zorunlu kılıyordu. Örneğin "kuşlar uçar" kuralı yazıldığında penguenler için ayrı bir kural eklemek gerekiyordu. Daha sonra yaralı kuşlar, yavru kuşlar veya farklı özel durumlar için yeni kurallar gerekiyordu. Bir süre sonra sistemin bilgi tabanı giderek büyüyor, kurallar birbirine karışıyor ve yönetilmesi zor hale geliyordu. İnsan uzmanların yıllarca biriktirdiği bilgileri kurallar halinde sisteme aktarmak hem çok pahalı hem de çok zaman alan bir süreçti.

Sorun yalnızca kural sayısının artması da değildi. Daha büyük problem, olası durumların sayısının inanılmaz bir hızla büyümesiydi. Buna yapay zekâ literatüründe "kombinatoryal patlama" adı verilir. *Basit bir örnek düşünelim. Bir sistemin yalnızca 10 farklı karar seçeneği varsa durum sayısı yönetilebilir olabilir. Ancak seçenek sayısı 20, 50 veya 100'e çıktığında olası kombinasyonların sayısı astronomik boyutlara ulaşır. Satranç bunun klasik örneklerinden biridir. Birkaç hamle sonrasını hesaplamaya çalıştığınızda karşılaşabileceğiniz durumların sayısı evrendeki atom sayısına yaklaşacak kadar büyüyebilir. Gerçek dünya ise satrançtan bile çok daha karmaşıktır.*

Bu nedenle sembolik sistemler küçük ve iyi tanımlanmış alanlarda başarılı olsalar da, gerçek hayatın belirsizlikleri karşısında zorlanmaya başladılar. İnsanlar yeni durumlarla karşılaştıklarında eksik bilgilerle karar verebilir, benzerliklerden yararlanabilir ve kuralları esnetebilirler. Sembolik sistemler ise genellikle yalnızca kendilerine açıkça öğretilen bilgileri kullanabiliyordu. Sistemin karşılaşmadığı yeni bir durum ortaya çıktığında çoğu zaman ne yapacağını bilemiyordu.

1980'lerin sonlarına doğru araştırmacılar önemli bir gerçeği fark etmeye başladılar. Belki de insan zekâsı yalnızca kurallar listesinden oluşmuyordu. Belki insanlar dünyayı milyonlarca kural ezberleyerek değil, deneyimlerden örüntüler öğrenerek anlıyorlardı. Bu farkındalık, sembolik yapay zekânın sınırlarını görünür hale getirdi. Aslında sembolik yaklaşım başarısız olduğu için değil, belirli bir noktaya kadar çok başarılı olduğu için tıkandı. Çünkü sistem büyüdükçe yeni kurallar eklemek zorlaşıyor, bakım maliyetleri artıyor ve gerçek dünyanın karmaşıklığı kurallarla yönetilemeyecek kadar büyük hale geliyordu.

Düşünme mimarisi açısından bakıldığında bu kriz son derece öğreticidir. Araştırmacılar ilk kez zekânın yalnızca kuralları uygulamak olmadığını anlamaya başladılar. Asıl mesele, sürekli

değişen ve eksik bilgiler içeren bir dünyada anlamlı davranış üretebilmekti. Sembolik yapay zekâ, düşünmenin mantıksal yönünü çok iyi açıklamıştı; ancak öğrenme, uyum sağlama ve genelleme gibi özellikleri açıklamakta zorlanıyordu. İşte bu tıkanma, daha sonra bağlantısalcı modellerin, sinir ağlarının ve nihayet derin öğrenmenin yükselişine zemin hazırlayacaktı. Çünkü araştırmacılar artık kuralları tek tek yazmak yerine, sistemlerin bu kuralları verilerden kendilerinin öğrenmesini sağlamaya çalışacaklardı.

Bu tıkanma, yalnızca teknik bir problem değildir; aynı zamanda düşünmenin doğasına dair temel bir sorunu açığa çıkarır. Eğer dünya sürekli yeni durumlar üretiyorsa, sabit kurallar üzerine kurulu bir sistem bu dünyayı nasıl temsil edebilir? Eğer bağlam sürekli değişiyorsa, önceden tanımlanmış çerçeve yapıları bu değişimi nasıl yakalayabilir? Bu sorular, sembolik yaklaşımın giderek daha fazla açıklama gücü kaybetmesine neden olur.

Sembolik yapay zekânın sınırları ortaya çıkmaya başladığında araştırmacılar yeni bir soru sormaya başladılar: Belki de insan zihni düşündüğümüz gibi kurallar kitabı şeklinde çalışmıyordu. Belki insanlar dünyayı binlerce kural ezberleyerek değil, deneyimlerden örüntüler öğrenerek anlıyordu. İşte bu düşünce, yapay zekâ araştırmalarını tamamen yeni bir yöne taşıdı. *Semboller, kurallar ve mantıksal çıkarımlar üzerine kurulu klasik yaklaşımın yerini yavaş yavaş Connectionism (Bağlantısalcılık) adı verilen yeni bir yaklaşım almaya başladı.*

Bağlantısalcılık, insan beyninden ilham alıyordu. Beyindeki tek bir nöronun çok akıllı olmadığı biliniyordu. Asıl önemli olan, milyarlarca nöronun birbirleriyle kurduğu bağlantılardı. Bu nedenle bağlantısalcı araştırmacılar, zekânın tek tek kurallarda veya sembollerde değil, çok sayıda basit birimin birlikte çalışmasında ortaya çıktığını savundular. Böylece zihin ilk kez bir kurallar kitabı olarak değil, birbirine bağlı dev bir ağ olarak düşünülmeye başlandı.

Bu yeni bakış açısında düşünme, önceden yazılmış mantık kurallarını uygulamak anlamına gelmiyordu. Bunun yerine sistem, çok sayıda örneği inceleyerek ilişkileri öğreniyordu. Örneğin sembolik bir sistemde bir kediyi tanımak için "dört ayağı vardır", "kulakları sivridir", "kuyruğu vardır" gibi kurallar yazmak gerekebilirdi. Bağlantısalcı sistemlerde ise binlerce kedi resmi gösterilir ve sistem zamanla kediyi diğer hayvanlardan ayıran örüntüleri kendi kendine öğrenirdi. Bu yaklaşım, gerçek dünyanın karmaşıklığıyla başa çıkmak için çok daha esnek bir yol sunuyordu.

Bağlantısalcı yaklaşımın en önemli iddiası şuydu: Bilgi tek tek sembollerin içinde saklanmaz; bağlantıların içinde saklanır. Bu fikir ilk başta biraz soyut görünebilir. Bunu bir şehir haritasına benzetebiliriz. Bir şehri şehir yapan şey yalnızca binalar değildir. Asıl önemli olan yolların, köprülerin ve bağlantıların nasıl düzenlendiğidir. Aynı şekilde bir sinir ağında da zekâ, tek tek düğümlerde değil, düğümler arasındaki bağlantıların gücünde ortaya çıkar.

Örneğin bir insanın "köpek" kavramını düşünelim. Sembolik yapay zekâda köpek, açıkça tanımlanmış bir sembol olarak temsil edilir. Sistemde "KÖPEK" adlı bir kavram vardır ve ona ilişkin kurallar yazılmıştır. Bağlantısalcı sistemlerde ise böyle tek bir sembol bulunmaz. Bunun yerine "köpek" bilgisi ağın birçok farklı yerine dağılmış durumdadır. Bazı bağlantılar köpeğin

görünüşüyle, bazıları hareketleriyle, bazıları sesleriyle, bazıları da diğer hayvanlarla olan ilişkileriyle ilgilidir. Yani köpek kavramı ağın tek bir noktasında değil, ağın genel yapısında saklanır.

Bu nedenle bağlantısalcı sistemlerde bilgiye ulaşmak da farklıdır. Bir dosya dolabından belirli bir klasörü çıkarmak gibi çalışmaz. Daha çok insan hafızasına benzer. Bir arkadaşınızın yüzünü gördüğünüzde onun adını, sesini, geçmiş anılarınızı ve kişiliğini aynı anda hatırlarsınız. Bunun nedeni, bu bilgilerin beyninizde tek bir yerde depolanmaması, çok sayıda bağlantı üzerinden birbirine bağlı olmasıdır. Bağlantısalcı araştırmacılar, insan zihninin de buna benzer çalıştığını ileri sürdüler.

Bu yaklaşım düşünmenin temsil modelini kökten değiştirdi. Sembolik sistemlerde bilgi açık, görünür ve ayrık durumdaydı. Kurallar okunabilir, incelenebilir ve değiştirilebilirdi. Bağlantısal sistemlerde ise bilgi ağın içine gömülüdür. Bir kavramın tam olarak nerede saklandığını göstermek çoğu zaman mümkün değildir. Çünkü bilgi ağın tamamına yayılmıştır. Bu nedenle bağlantısalcı sistemler bazen daha güçlü öğrenebilirken, aynı zamanda açıklanması daha zor sistemler haline gelirler.

Bugün kullandığımız modern yapay zekâ sistemleri de büyük ölçüde bu bağlantısalcı geleneğin devamıdır. ChatGPT, Gemini, Claude ve diğer büyük dil modelleri sembolik kurallar kitapları kullanmazlar. Onların bilgisi milyarlarca bağlantı ağırlığının içine dağılmış durumdadır. Bir kelimenin, bir kavramın veya bir fikrin tek bir adresi yoktur. Anlam, ağın genel organizasyonundan doğar. Bu nedenle modern yapay zekâ, aslında sembolik dönemin "bilgiyi kurallarla temsil etme" fikrinden uzaklaşarak, bağlantısalcılığın "bilgiyi ilişkiler içinde temsil etme" fikrine dayanır.

Düşünme mimarisi açısından bakıldığında bu dönüşüm son derece önemlidir. Çünkü ilk kez düşünme, semboller üzerinde çalışan bir mantık makinesi olarak değil, çok sayıda basit unsurun etkileşiminden doğan bir süreç olarak görülmeye başlanmıştır. Başka bir deyişle, zekâ artık kuralların sonucu değil, ağların sonucu olarak düşünülmektedir. Bu fikir daha sonra Geoffrey Hinton ve arkadaşlarının çalışmalarıyla güçlenecek, derin öğrenme devrimine ve günümüzdeki büyük dil modellerine kadar uzanan yolu açacaktır.

David Rumelhart ve James McClelland'ın çalışmaları, bağlantısalcı yaklaşımı yalnızca bir fikir olmaktan çıkarıp daha sistemli ve matematiksel bir çerçeveye oturtur. Özellikle 1980'lerde geliştirdikleri paralel dağıtık işleme (Parallel Distributed Processing – PDP) modeli, öğrenmenin nasıl gerçekleştiğine dair o zamana kadar baskın olan düşünceyi kökten değiştirir. Geleneksel yaklaşımda öğrenme, çoğunlukla açık kuralların sisteme tek tek öğretilmesi şeklinde düşünülüyordu. Yani bir sisteme "şu durumda şunu yap" gibi net talimatlar veriliyordu. *Ancak Rumelhart ve McClelland, bunun insan öğrenmesini açıklamakta yetersiz kaldığını gösterdiler.*

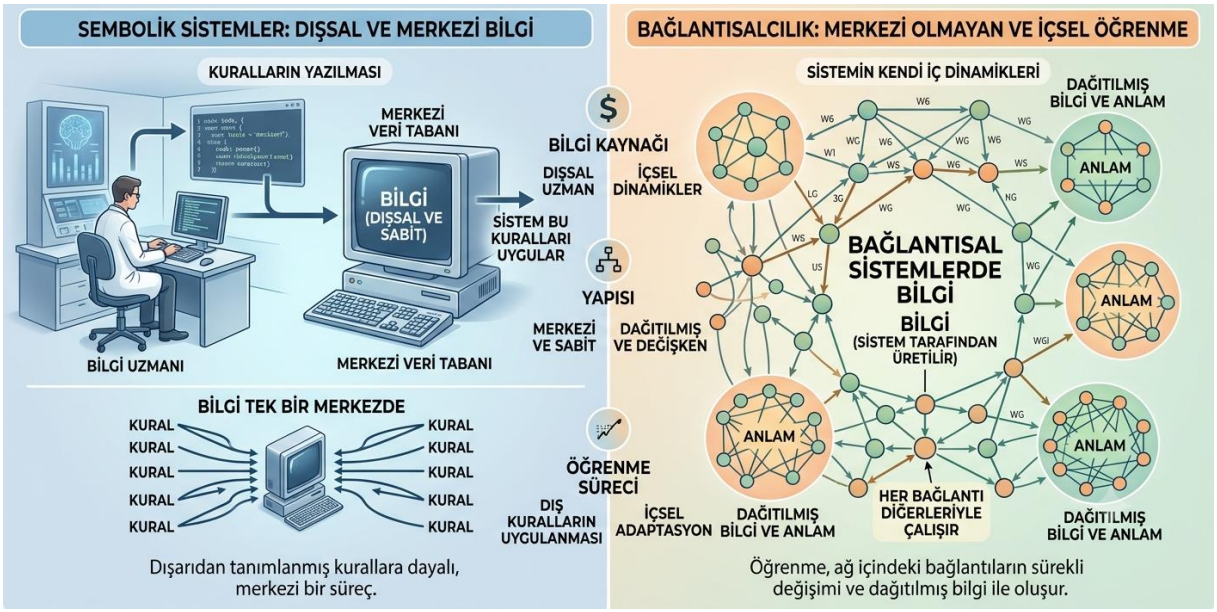
Onlara göre öğrenme, açık kuralların yazılmasıyla değil, çok sayıda örnekten gelen küçük sinyallerin birikmesiyle gerçekleşir. Sistem bir problemi çözmeyi doğrudan "öğretilmiş kurallar" sayesinde değil, defalarca karşılaştığı örneklerden yavaş yavaş öğrenir. Örneğin bir çocuk

geçmiş zaman fiillerini öğrenirken her fiil için ayrı bir kural ezberlemez. Bunun yerine birçok örnek görür, hatalar yapar, düzeltmeler alır ve zamanla doğru yapıyı sezgisel olarak öğrenir. PDP modelleri de tam olarak bu süreci taklit eder.

Bu yaklaşımda en önemli noktalardan biri şudur: sistemin içinde açık, okunabilir kurallar bulunmaz. Yani "eğer A ise B yap" gibi insan tarafından yazılmış kurallar yoktur. Bunun yerine öğrenme, sayısal değerlerin —yani bağlantı ağırlıklarının— sürekli olarak güncellenmesiyle gerçekleşir. Her yeni örnek, sistemde çok küçük bir değişikliğe yol açar. Bu küçük değişiklikler zaman içinde birikir ve sistemin davranışı şekillenir. Sonuçta ortaya çıkan bilgi, tek bir yerde duran açık bir kural değil, tüm ağın içine dağılmış bir yapıdır.

Bu durum düşünme kavramı açısından çok önemli bir kırılma yaratır. Çünkü artık zihin, dışarıdan programlanan bir sistem olarak değil, içeriden eğitilen bir sistem olarak görülmeye başlanır. Sembolik yapay zekâda insanlar sistemi adım adım programlar, yani zekâ dışarıdan inşa edilirdi. Rumelhart ve McClelland'ın yaklaşımında ise sistem kendi davranışını deneyim yoluyla geliştirir. Bu, mühendislik açısından bakıldığında kontrol edilen bir makineden çok, öğrenen bir organizmaya benzer.

Şekil 4. Yapay Zekada İki Yaklaşım: Sembolik Sistemler ve Bağlantısalcılık



Bağlantısalcılığın en önemli katkılarından biri de öğrenmenin merkezi olmayan (decentralized) bir süreç olduğunu açık biçimde göstermesidir. Sembolik sistemlerde bilgi genellikle dışarıdan tanımlanır. Yani bir uzman gelir, kuralları yazar ve sistem bu kuralları uygular. Bu nedenle bilgi dışsal ve sabittir. Bağlantısal sistemlerde ise bilgi dışarıdan verilmez; sistemin kendi iç dinamikleri tarafından üretilir. Her bağlantı, diğer bağlantılarla birlikte çalışarak anlamı oluşturur. Böylece bilgi tek bir merkezde değil, tüm sistemin içinde dağıtılmış olarak bulunur.

Bu bakış açısı, düşünmenin mühendislik modelini kökten değiştirir. Artık tasarımcıların görevi tek tek kurallar yazmak değildir. Çünkü böyle bir yaklaşım hem kırılgandır hem de ölçeklenmesi zordur. Onun yerine yapılması gereken şey, sistemin verilerden öğrenmesini sağlayacak uygun

bir yapı kurmaktır. Yani önemli olan “ne yapılacağını söylemek” değil, sistemin “kendi kendine nasıl öğreneceğini” tasarlamaktır. Bu, mühendislik açısından çok daha soyut ama aynı zamanda çok daha güçlü bir yaklaşımdır.

Bu dönüşüm aynı zamanda modern yapay zekâ anlayışının temelini oluşturur. Bugün büyük dil modelleri ya da derin öğrenme sistemleri açık kurallar yazılarak değil, büyük veri kümeleri üzerinde eğitilerek geliştirilir. Sistemlere dil kuralları tek tek öğretilmez; bunun yerine milyonlarca örnek gösterilir ve sistem bu örneklerden istatistiksel ilişkileri öğrenir. Bu süreç, Rumelhart ve McClelland’ın ortaya koyduğu fikrin çok daha büyük ölçekli bir versiyonu olarak görülebilir.

Sonuç olarak Bağlantısalcılık, düşünme anlayışını “programlama” paradigmasından “öğrenme” paradigmasına taşımıştır. Artık mesele bir sistemi dışarıdan kontrol etmek değil, sistemin içinden anlam üretmesini sağlamaktır. Bu nedenle bu yaklaşım, sadece yapay zekâ tarihinde değil, genel olarak zihnin nasıl çalıştığına dair anlayışımızda da derin bir dönüşüm yaratmıştır.

Bu dönüşümün en güçlü temsilcilerinden biri Geoffrey Hinton’dur. Hinton’un çalışmaları, bağlantısal sistemleri modern derin öğrenmenin temel yapı taşlarına dönüştürür. Onun yaklaşımında bilgi, sabit semboller veya açık kurallar şeklinde değil, ağırlık matrisleri içinde kodlanır. Bu ağırlıklar, sistemin deneyimle birlikte değişen içsel yapısını temsil eder.

Hinton’un katkısı, öğrenmenin doğasını yeniden tanımlamasıdır. Artık bilgi, önceden yazılmış kuralların uygulanması değil, veriden türeyen istatistiksel bir yapıdır. Sistem, örnekleri gördükçe içsel parametrelerini günceller ve bu güncellemeler zamanla karmaşık temsiller oluşturur. Bu temsiller, açıkça yazılmamış olsa bile güçlü genelleme yetenekleri üretir.

Bu yaklaşım, sembolik YZ’nin karşılaştığı temsil krizine doğrudan bir yanıt niteliğindedir. Çünkü artık dünya, tek tek kurallarla değil, örneklerden öğrenilen sürekli bir dağılım olarak modellenir. Bu model, belirsizliği daha doğal bir şekilde ele alır ve istisnaları özel durumlar olarak değil, öğrenme sürecinin bir parçası olarak kabul eder.

Bağlantısalcılık ve derin öğrenme yaklaşımı, düşünme mimarisinde yeni bir katman oluşturur. Bu katmanda zihin, açık kurallarla çalışan bir işlem sistemi değil, sürekli ayarlanan bir parametre uzayıdır. Bu parametreler, çevreden gelen verilerle şekillenir ve sistemin davranışını belirler. Böylece düşünme, açık programlamadan çok, sürekli optimizasyon süreci haline gelir.

Bu dönüşüm, aynı zamanda zihnin doğasına dair daha geniş bir soruyu gündeme getirir. Eğer düşünme sembollerle değil de ağırlıklarla temsil ediliyorsa, o zaman anlam nerede ortaya çıkar? Bu soru, bağlantısal yaklaşımın en derin felsefi problemlerinden biridir. Çünkü burada anlam, açık bir yapıda değil, sistemin tamamına dağılmış bir özellik olarak ortaya çıkar.

Sembolik YZ’nin tıkanması, bu nedenle yalnızca bir başarısızlık değil, aynı zamanda yeni bir paradigmanın doğuşudur. Temsil problemi, yerini öğrenme problemine bırakır. Önceden

tanımlanmış kuralların yerini, veriden türeyen yapılar alır. Açık sembollerin yerini, kapalı ama güçlü temsiller alır.

Bu kriz aynı zamanda düşünme mimarisinin yönünü de değiştirir. Artık zihin, tanımlanmış kuralların uygulandığı bir sistem değil, sürekli öğrenen ve kendini yeniden şekillendiren bir yapı olarak görülür. Bu yapı, sembolik dünyanın katı sınırlarını aşarak daha esnek, daha ölçeklenebilir ve daha uyumlu bir mimariye doğru evrilir.

Bu evrim, modern yapay zekânın temelini oluşturur. Bağlantısalcılık ve Hinton'un çalışmaları, düşünmenin artık yalnızca tasarlanan değil, aynı zamanda öğrenen bir sistem olduğunu gösterir. Bu noktadan itibaren düşünme, sabit kuralların uygulanması değil, sürekli değişen bir temsil alanının dinamik olarak şekillendirilmesidir.

Kaynaklar

Ashby, W. R. (1956). *An introduction to cybernetics*. Chapman & Hall.

Chomsky, N. (1957). *Syntactic structures*. Mouton.

Chomsky, N. (1965). *Aspects of the theory of syntax*. MIT Press.

Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition, 28*(1–2), 3–71.

Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1). MIT Press.

Karpathy, A. (2017). *Software 2.0*. https://karpathy.medium.com/software-2-0-a64152b37c35

Karpathy, A. (2015). *The unreasonable effectiveness of recurrent neural networks*. https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63*(2), 81–97.

Minsky, M. (1974). A framework for representing knowledge. In P. H. Winston (Ed.), *The psychology of computer vision*. McGraw-Hill.

Minsky, M. (1986). *The society of mind*. Simon & Schuster.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Prentice-Hall.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533–536.

Simon, H. A. (1957). *Models of man: Social and rational*. Wiley.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind, 59*(236), 433–460.

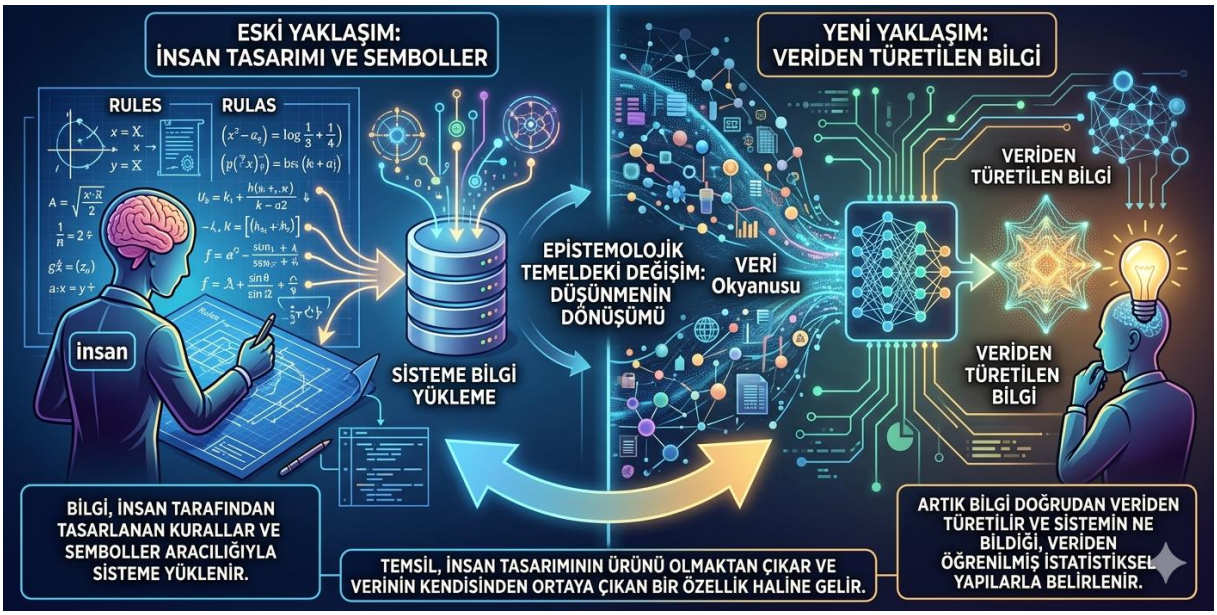
Wiener, N. (1948). *Cybernetics: Or control and communication in the animal and the machine*. MIT Press.

6. MODERN KIRILMA: DERİN ÖĞRENME

Sembolik yapay zekânın temsil krizinin ardından ortaya çıkan bağlantısal yaklaşım, uzun süre boyunca daha çok teorik ve sınırlı ölçekli sistemler düzeyinde kalır. Ancak 21. yüzyıla girildiğinde bu tabloyu kökten değiştiren bir birleşim gerçekleşir: hesaplama gücündeki dramatik artış, devasa veri setlerinin birikimi ve öğrenme algoritmalarındaki ilerlemeler. Bu üçlü birleşim, düşünme mimarisinde daha önce benzeri görülmemiş bir kırılma yaratır. Artık mesele yalnızca zihni modellemek değil, aynı zamanda onu büyük ölçekli veriden doğrudan öğrenen sistemler aracılığıyla yeniden inşa etmektir. Bu dönem, derin öğrenmenin doğuşudur.

Bu kırılmanın en temel özelliği, bilginin üretim biçimindeki radikal değişimdir. Daha önceki yaklaşımlarda bilgi, insan tarafından tasarlanan kurallar ve semboller aracılığıyla sisteme yüklenirken, artık bilgi doğrudan veriden türetilir. Bu dönüşüm, yalnızca teknik bir değişim değildir; düşünmenin epistemolojik temelini değiştirir. Çünkü artık sistemin neyi bildiği, önceden yazılmış kurallarla değil, veriden öğrenilmiş istatistiksel yapılarla belirlenir. Böylece temsil, insan tasarımının ürünü olmaktan çıkar ve verinin kendisinden ortaya çıkan bir özellik haline gelir.

Şekil 5. Bilgi Üretiminde Epistemolojik Dönüşüm



Büyük veri çağının ortaya çıkışı bu dönüşümün temel zemini olur. İnternetin yaygınlaşması, dijital etkileşimlerin artması ve sensör teknolojilerinin gelişmesiyle birlikte insanlık daha önce hiç olmadığı kadar büyük bir veri üretmeye başlar. Bu veri yığını, geleneksel sembolik sistemlerin işleyemeyeceği kadar karmaşıktır. Ancak aynı zamanda bu karmaşıklık, yeni bir fırsat alanı yaratır: eğer yeterince veri varsa, açık kurallar olmadan da öğrenme mümkün olabilir. Bu fikir, derin öğrenmenin temel varsayımını oluşturur.

Bu yeni yaklaşımda temsil anlayışı kökten değişir. Sembolik sistemlerde temsil açık, ayrık ve insan tarafından tanımlanmışken, derin öğrenmede temsil örtük, sürekli ve veri tarafından şekillendirilmiştir. El yapımı kuralların yerini, optimizasyon süreçleri alır. Sistem, veriyi analiz eder, hatalarını ölçer ve bu hataları azaltacak şekilde içsel parametrelerini günceller. Böylece bilgi, tasarlanan bir yapı olmaktan çıkar ve öğrenilen bir yapı haline gelir.

Derin sinir ağlarının yükselişi bu dönüşümün en görünür sonucudur. Çok katmanlı bu yapılar, basit dönüşümlerin üst üste eklenmesiyle giderek daha karmaşık temsiller üretir. İlk katmanlar genellikle düşük seviyeli özellikleri yakalarken, daha derin katmanlar daha soyut temsiller oluşturur. Bu hiyerarşik yapı, düşünmenin katmanlı doğasına dair güçlü bir analoji sunar. Çünkü burada bilgi, düz bir yüzeyde değil, derinlikli bir temsil uzayında şekillenir.

Derin ağların en önemli özelliği, öğrenmenin aşamalı bir temsil dönüşümü olarak gerçekleşmesidir. Sistem, ham veriyi alır ve bunu giderek daha anlamlı temsillere dönüştürür. Bu dönüşüm süreci, insan algısındaki soyutlama süreçlerine benzer bir yapı sergiler. Görsel sistemlerde kenarlar, şekiller ve nesneler; dil sistemlerinde kelimeler, cümle yapıları ve anlamlar bu katmanlar üzerinden inşa edilir. Böylece düşünme, tek bir işlem değil, çok katmanlı bir dönüşüm zinciri olarak yeniden tanımlanır.

Ancak derin öğrenmenin gerçek kırılma noktası yalnızca katmanlı yapılar değildir. Asıl devrim, bu yapıların ölçeklenebilir hale gelmesidir. Büyük veri ve güçlü hesaplama kaynakları birleştiğinde, sistemler daha önce mümkün olmayan düzeyde genelleme yetenekleri sergilemeye başlar. Bu durum, temsilin doğasına dair yeni bir soruyu gündeme getirir: anlam, açık kurallar olmadan da ortaya çıkabilir mi?

Bu sorunun cevabı, attention (dikkat) mekanizması ve transformer (dönüştürücü) mimarisinin ortaya çıkmasıyla birlikte köklü bir şekilde değişir. Geleneksel sinir ağlarında bilgi genellikle belirli ve sabit bir yapı içinde işlenir. Yani sistem, girdileri çoğu zaman aynı türden parçalar gibi görür ve bunları önceden tanımlanmış katmanlar üzerinden sırayla işler. Bu yaklaşımda her bilginin ağırlığı büyük ölçüde modelin öğrenme sürecinde sabitlenir ve bağlama göre anlık olarak değişmez.

Attention (dikkat) mekanizması ise bu tabloyu önemli ölçüde dönüştürür. Bu mekanizma sayesinde sistem, gelen tüm bilgileri aynı düzeyde ve eşit önemde değerlendirmek zorunda kalmaz. Bunun yerine, her yeni durumda hangi bilginin daha önemli olduğuna dinamik olarak karar verir. Yani model, “her şey eşit derecede önemli” yaklaşımından uzaklaşır ve “bu bağlamda ne önemliyse ona daha çok odaklan” ilkesine geçer. Böylece bilgi, artık sadece var olan bir veri parçası değil, içinde bulunduğu bağlama göre anlam kazanan bir öğe haline gelir.

Attention (dikkat) fikrinin düşünme mimarisi açısından en önemli sonucu, bilginin artık sabit bir kutu içinde değil, sürekli değişen ilişkiler ağı içinde temsil edilmesidir. Burada her bilgi parçası tek başına anlamlı bir nesne olmaktan çıkar; diğer bilgi parçalarıyla kurduğu ilişkiler üzerinden anlam kazanır. Örneğin bir kelime, yalnızca kendi tanımıyla değil, cümledeki diğer

kelimelerle olan ilişkisine göre yorumlanır. Bu durum, düşünmenin daha ilişkisel ve bağlama duyarlı bir süreç olarak yeniden anlaşılmasını sağlar.

Transformer (dönüştürücü) mimarisi ise bu yaklaşımı daha ileri bir seviyeye taşır. Özellikle self-attention (öz-dikkat) adı verilen mekanizma sayesinde sistem, bir dizide bulunan her bir öğeyi aynı anda diğer tüm öğelerle karşılaştırır ve aralarındaki ilişkileri hesaplar. Bunun sonucunda tek bir noktaya değil, tüm diziyi kapsayan geniş bir bağlam haritasına ulaşılır. Bu yapı, özellikle dil gibi uzun mesafeli bağımlılıkların kritik olduğu alanlarda büyük bir dönüşüm yaratır. Çünkü artık model sadece yakınındaki kelimelere veya yerel örüntülere değil, metnin tamamına yayılan ilişkilere aynı anda bakabilir ve daha tutarlı, daha bütüncül bir temsil oluşturabilir.

Bu gelişme, düşünme mimarisinde yeni bir seviyeye işaret eder: bağlam artık yardımcı bir unsur değil, doğrudan hesaplamanın merkezidir. Her çıktı, tüm giriş uzayının yeniden ağırlıklandırılmasıyla üretilir. Böylece düşünme, sabit kuralların uygulanması değil, dinamik bağlam hesaplaması haline gelir.

Bu modern kırılma, düşünme tarihindeki önceki tüm aşamaları yeniden yorumlar. Sembolik YZ'nin kuralları yerini öğrenilmiş temsillere bırakır. Bağlantısalcı yaklaşımın dağıtık yapısı, daha büyük ölçeklerde stabil hale gelir. Sibernetik geri besleme fikri, optimizasyon süreçleri içinde yeniden ortaya çıkar. Ve en önemlisi, zihnin sistem olarak görülmesi fikri artık teorik bir öneri değil, çalışan bir mühendislik gerçekliğine dönüşür.

Derin öğrenme ile birlikte düşünme, ilk kez bu kadar geniş ölçekte "çalışan bir sistem" haline gelir. Bu sistem, insan tarafından açıkça programlanmamış olsa da, veri ve hesaplama gücü aracılığıyla kendi içsel temsillerini oluşturur. Böylece zihnin mühendislik tarihi yeni bir aşamaya geçer: artık sistemler yalnızca tasarlanan değil, aynı zamanda ölçekle öğrenen yapılardır.

Kaynaklar

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(8), 1798–1828.

Brown, T. B., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Hinton, G. E. (2007). Learning multiple layers of representation. *Trends in Cognitive Sciences, 11*(10), 428–434.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*(7), 1527–1554.

Kaplan, J., et al. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533–536.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences, 11*(1), 1–23.

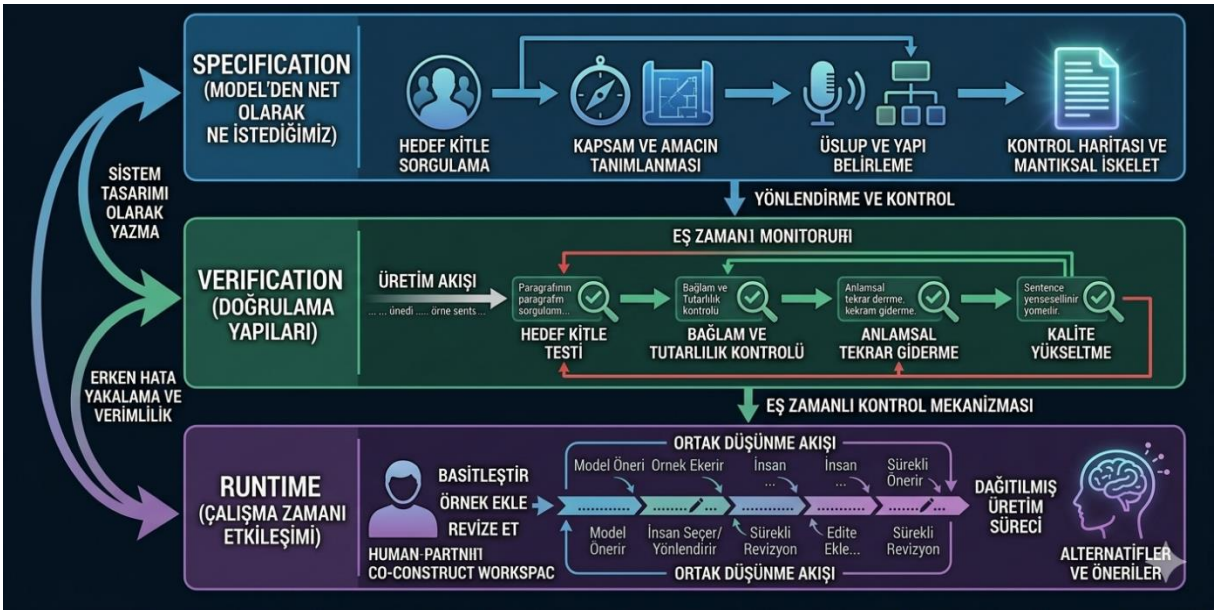
7. KARPATHY VE METİN ÜRETİM SİSTEMİ

Derin öğrenme ve transformer mimarilerinin ortaya çıkışıyla birlikte düşünme artık yalnızca teorik bir model olmaktan çıkar ve gerçek zamanlı çalışan üretim sistemlerine dönüşür. Ancak bu dönüşümün en kritik aşaması, yalnızca modelin ne yaptığı değil, insanın bu sistemlerle nasıl çalıştığı sorusudur. Çünkü yapay zekânın gelişimi yalnızca algoritmaların evrimi değildir; aynı zamanda insanın problem çözme, yazma, analiz etme ve düşünme yöntemlerinin de kökten değişmesidir. Bu noktada Andrej Karpathy'nin önerdiği çerçeve, yalnızca teknik bir yaklaşım değil, aynı zamanda yeni bir düşünme ergonomisidir. Bu ergonomi, metin üretimini üç temel katmanda yeniden organize eder: spesifikasyon, doğrulama ve çalışma zamanı.

Geleneksel yaklaşımda metin üretimi çoğunlukla lineer bir süreçtir. Kullanıcı bir fikirle başlar, bu fikri zihninde geliştirir, ardından onu yazıya döker ve son olarak düzenler. Akademik yazımda bu süreç daha da katı bir forma sahiptir: önce taslak hazırlanır, sonra kaynaklar eklenir, ardından revizyon yapılır ve en sonunda metin tamamlanır. Kurumsal yazımda ise süreç daha da parçalıdır; fikirler çoğunlukla e-posta zincirlerinde, notlarda veya ayrı dokümanlarda dağınık şekilde oluşur ve daha sonra birleştirilir. *Bu modelde düşünme ile yazma arasında net bir ayrım vardır. Düşünme içsel bir süreç, yazma ise dışsal bir çıktı olarak görülür.*

Bu geleneksel modelin temel problemi şudur: üretim süreci ile doğrulama süreci birbirinden ayrılmıştır. İnsan önce üretir, sonra kontrol eder. Bu durum özellikle karmaşık metinlerde ciddi verimsizlik yaratır. Çünkü hatalar genellikle üretim aşamasında fark edilmez, ancak sonradan geri dönülerek düzeltilir. Bu da tekrar eden döngüler, zihinsel yük ve zaman kaybı yaratır. Karpathy'nin yaklaşımı tam olarak bu ayrımı sorgular ve üretimi tek bir lineer süreç olarak değil, sürekli etkileşim halinde olan bir sistem olarak yeniden tanımlar.

Şekil 6. Karpathy ve Katmanlı Düşünme Mimarisi



Bu yeni çerçevede ilk katman "specification (modelden net olarak ne istediğimiz)" yani modelden net olarak ne istediğimizin açık, ayrıntılı ve mümkün olduğunca kesin bir şekilde tanımlandığı aşamadır. Buradaki "modelden net olarak ne istediğimiz", sadece genel bir niyet değil; üretilecek çıktının ne olacağını baştan sona tarif eden bir kontrol haritası gibidir. Geleneksel yazım süreçlerinde bu aşama çoğu zaman bulanık kalır. İnsanlar genellikle "bir konu seçip yazmaya başlama" refleksiyle hareket eder. Yani önce üretim başlar, sonra anlam netleşir. Bu da çoğu zaman dağınık, hedefi belirsiz ve sonradan çok revizyon gerektiren metinler üretir.

Karpathy'nin yaklaşımında ise bu durum ciddi bir verimsizlik olarak görülür. Çünkü modelden net olarak ne istediğimiz açık değilse, üzerine inşa edilen tüm üretim süreci de aynı derecede belirsiz olur. Bir binanın temeli doğru çizilmeden kat çıkmaya benzer: üst katlar ne kadar iyi yapılırsa yapılsın, yapı genel olarak dengesiz kalır. Bu nedenle specification (modelden net olarak ne istediğimiz) aşaması, sadece bir hazırlık değil, tüm üretim sisteminin davranışını belirleyen temel kontrol katmanıdır.

Bu aşamada amaç, üretilecek metnin sınırlarını, amacını, hedef kitlesini, tonunu ve yapısını açıkça tanımlamaktır. Örneğin bir metin "yapay zekâ hakkında yazı" olarak bırakılırsa bu çok geniş ve kontrolsüz bir alan yaratır. Ancak belirleme doğru yapılırsa bu ifade çok daha net hale gelir: "Yapay zekânın akademik yazım süreçlerini nasıl değiştirdiğini, teknik ayrıntıya girmeden, lisansüstü öğrenciler için açıklayan, örneklerle desteklenmiş, sade bir bölüm yaz." Bu ikinci durumda modelden net olarak ne istediğimiz artık belirsiz değildir; oldukça yönlendirilmiş bir üretim alanı tanımlanmıştır.

Örneğin geleneksel bir yaklaşımda bir akademisyen şöyle çalışır: "Yapay zekâ hakkında bir bölüm yazacağım" diyerek sürece başlar. Burada henüz modelden net olarak ne istediğimiz tanımlanmamıştır. Bu nedenle yazım süreci ilerledikçe konular dağılır, bazı yerlerde teknik detaylara gereksiz şekilde girilir, bazı yerlerde ise yüzeysel kalınır.

Karpathy'nin çerçevesinde ise aynı süreç çok daha ayrıntılı bir specifikasyon (modelden net olarak ne istediğimiz) aşamasıyla başlar. Örneğin bu aşamada şu sorular açıkça yanıtlanır:

– Metnin amacı nedir?
Örneğin: "Akademisyenlere transformer mimarisinin yazım süreçlerini nasıl değiştirdiğini açıklamak."

– Hedef kitle kimdir?
Örneğin: "Makine öğrenmesi bilen ama üretim sistemleri tasarımına yeni başlayan araştırmacılar."

– Kapsam nedir?
Örneğin: "Sadece specification, decomposition ve verification katmanları anlatılacak; reinforcement learning veya düşük seviye optimizasyon detaylarına girilmeyecek."

– Ton nasıl olmalıdır?
Örneğin: "Akademik ama sade; gereksiz jargon içermeyen, bol örnekli."

– Yapı nasıl olmalıdır?
Örneğin: "Önce kavramsal çerçeve, sonra geleneksel yöntemle karşılaştırma, ardından örnek senaryo ve en son pratik çıkarımlar."

Bu şekilde modelden net olarak ne istediğimiz adım adım tanımlandığında, artık yazım süreci rastlantısal bir süreç olmaktan çıkar. Bunun yerine kontrollü ve yönlendirilmiş bir üretim sürecine dönüşür.

Bu aşama aynı zamanda düşünmenin kendisini de yeniden yapılandırır. Çünkü burada yazıdan önce zihinsel model oluşturulur. Örneğin "yapay zekâ hakkında yazmak" gibi genel bir düşünce, modelden net olarak ne istediğimiz aşamasında parçalanır ve daha küçük, daha net alt sorulara ayrılır. Bu da düşünmeyi daha sistematik hale getirir. Artık zihin "ne yazacağım?" sorusuyla değil, "tam olarak hangi problemi, hangi sınırlar içinde, hangi örneklerle anlatacağım?" sorusuyla çalışır.

Örneğin bir eğitim senaryosu düşünelim. Geleneksel yaklaşımda bir öğretmen "öğrencilere yapay zekâyı anlatacağım" diyerek derse girer. Ancak belirleme (modelden net olarak ne istediğimiz) yapılmış bir yaklaşımda bu ifade şöyle parçalanır:

– Öğrenciler hangi seviyede? (örneğin: lisans 2. sınıf)
– Hangi kavramlar biliniyor? (örneğin: temel lineer cebir)
– Hangi kavramlar yeni verilecek? (örneğin: dikkat mekanizması)
– Hangi örnek kullanılacak? (örneğin: metin çeviri sistemi)

Bu netlik sayesinde anlatım sadece daha düzenli değil, aynı zamanda daha öngörülebilir ve ölçülebilir hale gelir. Sonuç olarak belirleme aşaması, sadece bir yazım ön adımı değil; tüm üretim sürecinin mantıksal iskeletini kuran temel katmandır. Bu katman ne kadar net olursa, üzerine inşa edilen tüm düşünme ve üretim süreci o kadar güçlü, tutarlı ve kontrollü olur.

İkinci katman “verification” yani doğrulama aşamasıdır. Burada “doğrulama”, sadece yazı bittikten sonra yapılan basit bir hata kontrolü değil, modelden net olarak ne istediğimiz doğrultusunda üretilen her parçanın sürekli olarak test edilmesi, sınanması ve yeniden değerlendirilmesi sürecidir. Geleneksel yazım süreçlerinde doğrulama genellikle sürecin en sonuna bırakılır. Yazar önce metni üretir, ardından geriye dönüp okur, yazım hatalarını düzeltir, bazı cümleleri yeniden yazar, mantık hatalarını gidermeye çalışır. Yani doğrulama burada ayrı bir “son kontrol adımı” gibi çalışır. Bu yaklaşımda problem şudur: hata, sistemin içine çoktan yerleşmiş olur ve ancak üretim bittikten sonra fark edilir.

Örneğin bir akademik makale yazıldığını düşünelim. Yazar önce giriş bölümünü, sonra literatür taramasını, sonra yöntemi ve sonuçları yazar. Ancak en sonunda geri dönüp baktığında şu sorunları fark eder: bazı kavramlar tutarsız kullanılmıştır (örneğin “model” bazen istatistiksel model, bazen yapay zekâ modeli anlamında kullanılmıştır), bazı argümanlar yeterince desteklenmemiştir veya bazı bölümler gereksiz yere uzamıştır. Bu durumda doğrulama süreci, zaten tamamlanmış bir yapıyı onarmaya çalışır. Bu da hem zaman kaybına hem de yapısal bozulmalara yol açar.

Karpathy’nin yaklaşımında ise doğrulama tamamen farklı bir şekilde konumlanır. Burada doğrulama, üretim bittikten sonra yapılan bir işlem değil, üretimle eş zamanlı çalışan bir kontrol mekanizmasıdır. Yani yazılan her cümle, her paragraf ve her bölüm, daha oluştuğu anda modelden net olarak ne istediğimiz ile karşılaştırılır ve sürekli olarak test edilir. Bu nedenle sistem, yalnızca “üreten” bir yapı değil, aynı zamanda sürekli “denetleyen” bir yapı haline gelir.

Bu yaklaşımda kullanıcı yalnızca üretici değil, aynı zamanda sürekli bir denetleyicidir. Yani yazı yazmak artık tek yönlü bir süreç değildir. Her üretim adımı, aynı anda bir doğrulama adımıyla birlikte çalışır. Örneğin bir paragraf yazıldığında hemen şu sorular devreye girer: “Bu paragraf gerçekten belirleme ile uyumlu mu?”, “Burada anlatılan fikir, hedef kitle için fazla teknik mi yoksa yeterince açık mı?”, “Bu örnek gerçekten ana argümanı destekliyor mu, yoksa sadece dolgu mu?”, “Bu cümle daha önce söylenen bir şeyi gereksiz şekilde tekrar ediyor mu?”

Örneğin bir metin yazıldığını düşünelim: “Transformer mimarisi dil modellerini geliştirmiştir.” Doğrulama katmanı hemen devreye girer ve şu tür sorular üretir: Bu ifade fazla genel mi? Hangi yönüyle geliştirmiştir? Performans mı, hız mı, bağlam anlama mı? Eğer bu sorulara cevap yoksa, cümle kabul edilmez ve yeniden yazılır. Örneğin daha doğru bir versiyon şu hale gelir: “Transformer mimarisi, uzun mesafeli bağımlılıkları daha iyi modelleyerek dil modellerinin bağlamı daha bütüncül anlamasını sağlamıştır.” Burada doğrulama, sadece hatayı bulmaz, aynı zamanda ifadenin kalitesini de yükseltir.

Başka bir örnekte, bir paragraf içinde aynı fikir iki kez farklı cümlelerle tekrar ediyorsa, geleneksel yaklaşımda bu durum ancak son okumada fark edilir. Ancak doğrulama yaklaşımında sistem bunu anında yakalar: “Bu cümle önceki cümlenin anlamsal tekrarımı?” Eğer evet ise ya birleştirilir ya da biri çıkarılır. Böylece metin şişmez, gereksiz tekrarlarla zayıflamaz.

Bir başka örnek: Diyelim ki bir eğitim metni yazılıyor ve hedef kitle “lisans öğrencileri” olarak belirlenmiş. Eğer metin içinde “ geriye doğru yayılım (backpropagation) türev zinciri üzerinden gradyanların hesaplanması” gibi aşırı teknik bir ifade geçerse, doğrulama katmanı bunu işaretler: “Bu ifade hedef kitle için fazla karmaşık.” Bunun yerine daha uygun bir açıklama önerilir: “Modelin hatalarından öğrenebilmesi için, hatanın geriye doğru yayılması süreci.” Böylece içerik, sürekli olarak hedef kitleye uygunluk açısından test edilir.

Bu sürekli doğrulama yaklaşımı, düşünme biçimini de kökten değiştirir. Artık düşünme ile doğrulama birbirinden ayrılmış iki aşama değildir. Geleneksel zihinsel modelde kişi önce düşünür, sonra yazar, sonra kontrol eder. Ancak bu yeni modelde düşünme süreci zaten doğrulama ile iç içedir. Yani kişi bir fikir üretirken aynı anda onun geçerliliğini de sorgular. “Bu argüman gerçekten tutarlı mı?”, “Bu örnek doğru bir genelleme yapıyor mu?”, “Bu ifade fazla iddialı mı?” gibi sorular düşünmenin doğal bir parçası haline gelir.

Örneğin bir araştırmacı “yapay zekâ eğitim sistemini dönüştürüyor” dediğinde, doğrulama süreci hemen devreye girer: Hangi yönüyle dönüştürüyor? Ölçülebilir bir kanıt var mı? Bu dönüşüm her eğitim seviyesinde geçerli mi? Eğer bu sorulara cevap verilemiyorsa, ifade ya daraltılır ya da yeniden yapılandırılır: “Yapay zekâ, özellikle kişiselleştirilmiş öğrenme sistemleri üzerinden eğitim süreçlerinin bazı yönlerini yeniden şekillendirmektedir.”

Sonuç olarak doğrulama katmanı, metni yazıldıktan sonra düzeltilen bir nesne olmaktan çıkarır ve onu sürekli olarak kontrol edilen, yaşayan bir yapı haline getirir. Bu yapı içinde üretim ve doğrulama birbirinden ayrılmaz; her kelime, her cümle ve her fikir aynı anda hem oluşturulur hem de test edilir.

Geleneksel yöntemle çalışan bir kişi genellikle şöyle bir süreç yaşar: önce uzun bir metin üretir, ardından geri dönüp “burada ne demek istemiştim” sorusunu sorar. Karpathy’nin önerdiği modelde ise bu problem ortadan kalkar, çünkü her üretim adımı aynı zamanda bir doğrulama adımıdır. Bu yaklaşım, bilişsel yükü azaltır ve hataların erken aşamada yakalanmasını sağlar.

Üçüncü katman “çalışma zamanı” yani sistemin aktif olarak çalıştığı, üretimin gerçekten gerçekleştiği aşamadır. Burada “çalışma zamanı”, sadece metnin yazıldığı anı değil, aynı zamanda model ile kullanıcı arasındaki canlı etkileşimin sürdüğü süreci ifade eder. Geleneksel yazım modelinde bu aşama oldukça basittir ve çoğu zaman tek yönlüdür: kullanıcı bir şey yazar, model bir çıktı üretir ve süreç büyük ölçüde biter. Yani bilgi akışı genellikle “insandan makineye” ve ardından “makineden insana” şeklinde iki adımlı ve kapalı bir döngü gibidir.

Örneğin klasik kullanımda bir kullanıcı şöyle yazar: “Yapay zekâ hakkında bir paragraf yaz.” Model bir paragraf üretir, kullanıcı okur ve ya kabul eder ya da küçük değişiklikler yapar. Bu süreçte etkileşim sınırlıdır; model üretir, kullanıcı değerlendirir. Arada gerçek anlamda ortak bir düşünme süreci yoktur.

Karpathy’nin çerçevesinde ise çalışma zamanı çok daha dinamik ve sürekli bir etkileşim alanıdır. Burada kullanıcı ve model arasında sürekli bir geri besleme döngüsü vardır. Yani süreç tek seferlik bir üretim değil, adım adım birlikte şekillenen bir düşünme akışıdır.

Örneğin kullanıcı bir cümle yazdığında model sadece devam ettirmez; aynı zamanda alternatifler üretir, farklı versiyonlar önerir ve kullanıcıya seçim imkânı sunar. Kullanıcı da bu öneriler arasından seçim yapar, bazılarını birleştirir veya tamamen yeniden yön verir. Bu şekilde metin, tek bir ham üretim değil, sürekli revize edilen bir yapı haline gelir.

Basit bir örnek düşünelim: Kullanıcı "yapay zekânın eğitim üzerindeki etkileri" hakkında bir paragraf yazmak istiyor. Geleneksel modelde tek bir paragraf üretilir. Çalışma zamanı yaklaşımında ise bu süreç şöyle gelişir:

– Model önce bir giriş paragrafı önerir.
– Kullanıcı der ki: "Bunu daha basit yap."
– Model aynı içeriği daha sade bir versiyona çevirir.
– Kullanıcı ekler: "Bir örnek ekleyelim."
– Model bir örnek üretir: örneğin kişiselleştirilmiş öğrenme sistemleri.
– Kullanıcı tekrar sorar: "Bunu akademik tona biraz daha yaklaştır."
– Model metni yeniden düzenler.

Bu süreçte metin, sabit bir çıktı olmaktan çıkar; sürekli birlikte inşa edilen (bir yapıya dönüşür.

Başka bir örnek: Bir araştırmacı makale yazarken "giriş bölümünü yaz" dediğinde model bir taslak üretir. Araştırmacı "burada çok genel olmuş" der. Model daha teknik bir versiyon üretir. Araştırmacı "bir referans tarzı açıklama ekle" der. Model bunu ekler. Sonra araştırmacı "burayı iki parçaya böl" der. Model yeniden yapılandırır. Böylece metin, tek bir üretim anından değil, çok sayıda küçük etkileşimden oluşur.

Bu yaklaşımda model artık sadece bir araç değildir. Aynı zamanda bir düşünme ortağı haline gelir. Yani kullanıcı yalnızca yazan kişi değildir; modelle birlikte düşünen, birlikte karar veren ve birlikte yapı kuran bir aktördür. Model ise yalnızca metin üreten bir sistem değil, aynı zamanda öneren, sorgulayan ve yönlendiren bir ikinci zihin gibi çalışır.

Örneğin kullanıcı bir fikir yazarken model şunu sorabilir: "Bu argümanı biraz daha somutlaştırmak ister misin?" veya "Bunu daha güçlü bir örnekle destekleyebiliriz." Bu tür müdahaleler, yazım sürecini tek yönlü bir üretimden çıkarır ve iki yönlü bir düşünme sürecine dönüştürür.

Bu durumda metin artık tek bir zihnin ürünü değildir. İnsan ve model arasında dağıtılmış bir üretim süreci ortaya çıkar. Yani fikirler, cümleler ve yapı tek bir noktadan değil, sürekli etkileşim içinde şekillenir. Örneğin bir blog yazısı düşünelim. Geleneksel modelde yazar oturur ve yazıyı tek başına üretir. Runtime (çalışma zamanı) yaklaşımında ise yazar ve model birlikte çalışır: biri fikir üretir, diğeri alternatif sunar, biri sadeleştirir, diğeri örnek ekler, biri yapı kurar, diğeri yapıyı optimize eder.

Bu durum düşünmenin doğasını da değiştirir. Artık düşünme sadece bireysel bir etkinlik değildir; etkileşimsel bir sistem haline gelir. Yani düşünme, tek bir zihnin içinde gerçekleşen kapalı bir süreç olmaktan çıkar ve iki taraflı, sürekli geri bildirimle gelişen bir yapı haline gelir.

Sonuç olarak çalışma zamanı, sadece metnin üretildiği an değil; düşünmenin, yazmanın ve karar vermenin birlikte gerçekleştiği canlı bir etkileşim alanıdır.

Karpathy'nin çerçevesinin en önemli katkılarından biri, bu üç katmanı birbirinden bağımsız aşamalar olarak değil, sürekli etkileşim halinde olan bir sistem olarak görmesidir. Spesifikasyon doğrulamayı yönlendirir, doğrulama spesifikasyonu günceller, çalışma zamanı ise her ikisini sürekli yeniden şekillendirir. Bu döngüsel yapı, klasik yazım süreçlerinin lineer doğasını ortadan kaldırır.

Bu çerçeve yalnızca teknik bir öneri değildir; aynı zamanda insanın yazma ve düşünme alışkanlıklarını yeniden düzenleyen bir modeldir. Geleneksel yazımda insanlar çoğu zaman "önce düşün, sonra yaz" yaklaşımını benimser. Ancak bu yaklaşımda düşünme ve yazma birbirinden ayrıdır. Karpathy'nin modelinde ise yazma, düşünmenin bir uzantısı değil, düşünmenin kendisidir. Çünkü spesifikasyon yazılırken düşünülür, doğrulama yapılırken düşünce test edilir, çalışma zamanı sırasında ise düşünce şekillenir.

Bu yaklaşımın daha ileri bir boyutu "iç diyalog mekanizması (inner dialogue mechanism / içsel konuşma mekanizması)"dır. Buradaki fikir şudur: modern büyük dil modelleri yalnızca tek adımda cevap veren sistemler değildir; aynı zamanda kendi içinde "düşünen", "kendine soru soran" ve "farklı seçenekleri deneyen" bir yapı gibi çalışabilir. Yani model, dışarıya verdiği cevaptan önce içeride bir tür iç konuşma, iç monolog yürütür.

Bu içsel süreçte adım adım düşünme zinciri, reflection (yansıtma / geri düşünme) ve self-critique (öz-eleştiri) gibi mekanizmalar devreye girer. Bunları basitçe şöyle düşünebiliriz: model önce problemi küçük parçalara böler, sonra her parçayı ayrı ayrı düşünür, sonra kendi ürettiği cevapları kontrol eder ve en sonunda en uygun olanı seçer. Yani tek bir "cevap üretme" işlemi yoktur; onun yerine içeride küçük bir düşünme toplantısı yapılır gibi bir süreç vardır.

Örneğin çok basit bir soru düşünelim: "Bu metni açıkla." Geleneksel kullanımda model hemen bir açıklama üretir ve tek bir çıktı verir. Süreç bu kadarla biter. Ama iç diyalog mekanizması olan bir yaklaşımda model önce kendi içinde şunu yapar:

– "Bu metnin ana fikri ne?"
– "Hangi kavramlar önemli?"
– "Bunu teknik mi anlatmalıyım, yoksa basit mi?"
– "Örnek vermek faydalı olur mu?"

Sonra farklı açıklama yolları üretir. Örneğin bir seçenek: çok teknik açıklama. İkinci seçenek: günlük dille açıklama. Üçüncü seçenek: örnekli açıklama. Daha sonra bu seçenekleri karşılaştırır:

– Teknik açıklama doğru ama zor olabilir
– Günlük dil açıklama anlaşılır ama yüzeysel kalabilir
– Örnekli açıklama hem anlaşılır hem açıklayıcı olabilir

Sonunda en uygun seçeneği seçer ve dışarıya tek bir cevap verir. Ama bu tek cevap, içeride birçok alternatifin tartışılıp elenmesiyle oluşmuştur.

Başka bir örnek: “Bu problemi çöz” komutu verildiğini düşünelim. Geleneksel model hemen bir çözüm verir. İç diyalog mekanizmasında ise model önce şöyle bir süreç yürütür:

– “Bu problem hangi kategoriye ait?” (matematik mi, mantık mı, istatistik mi?)
– “Hangi yöntemler kullanılabilir?”
– “Birden fazla çözüm yolu var mı?”

Sonra örneğin iki farklı yol üretir: birinci yol kısa ama riskli, ikinci yol daha uzun ama güvenilir. Ardından bu yolları test eder:

– “Birinci yol hızlı ama hata riski var mı?”
– “İkinci yol daha sağlam ama gereksiz karmaşık mı?”

Bu değerlendirmeden sonra en iyi görünen yolu seçer ve sonucu verir.

Örneğin bir matematik problemi düşünelim: “Bir tren 2 saatte 120 km gidiyor, hız nedir?” Model içeride şunu yapar:

– “Hız = yol / zaman”
– “120 / 2 = 60”
– “Birim doğru mu? km/saat olmalı”
– “Sonuç tutarlı mı?”

Bu küçük kontrol adımları, dışarıdan görünmeyen bir iç doğrulama süreci oluşturur.

Bir başka örnek: bir metin yazma görevi verildiğinde model şunu yapar:

– “Bu paragraf giriş mi olmalı, yoksa açıklama mı?”
– “Bu cümle fazla uzun mu?”
– “Bu ifade hedef kitle için uygun mu?”
– “Daha basit bir alternatif var mı?”

Sonra birkaç versiyon üretir ve içinden en iyi olanı seçer.

Bu nedenle Karpathy’nin yaklaşımında model, sadece dışarıdan gelen komutlara tepki veren bir sistem değildir. Aynı zamanda kendi içinde sürekli konuşan, alternatif üreten ve kendini kontrol eden bir düşünme sistemi gibi davranır. Bu iç diyalog sayesinde tek bir cevap bile aslında çok sayıda küçük düşünme adımının sonucudur. Sonuç olarak iç diyalog mekanizması, modeli “tek adımda cevap veren bir araç” olmaktan çıkarır ve onu “kendi içinde düşünen, sorgulayan ve seçen bir sistem” haline getirir.

İnsan açısından bakıldığında bu durum da oldukça benzer ve önemli bir dönüşüm yaratır. Geleneksel yazım sürecinde kişi genellikle tek bir düşünce hattı üzerinden ilerler. Yani önce bir fikir belirlenir, sonra bu fikir cümlelere dökülür ve yazı büyük ölçüde bu doğrultuda tamamlanır.

Bu süreçte düşünme çoğu zaman “ileri doğru akan bir nehir” gibidir: geri dönmek, farklı bir yol denemek ya hiç yapılmaz ya da en sona bırakılır.

Ancak iç diyalog yaklaşımında düşünme artık tek bir çizgi değil, sürekli dallanan bir süreç haline gelir. Kişi yazarken aynı anda farklı seçenekleri üretir ve bu seçenekleri zihninde karşılaştırır. Örneğin bir paragraf yazarken sadece “ne yazıyorum?” sorusu değil, aynı anda birçok küçük soru aktif hale gelir: “Bunu daha basit nasıl söyleyebilirim?”, “Bu cümle fazla akademik mi kaldı?”, “Daha güçlü bir örnek kullanabilir miyim?”, “Bu argüman gerçekten ikna edici mi yoksa zayıf mı?”, “Okuyucu bunu ilk okumada anlayabilir mi?”

Örneğin bir yazar şu cümleyi yazdığını düşünelim: “Yapay zekâ modern dünyayı dönüştürmektedir.” İç diyalog sürecinde zihin hemen alternatifler üretir:

– “Bu çok genel, biraz açmalıyım.”
– “Hangi açıdan dönüştürüyor?”
– “Ekonomi mi, eğitim mi, günlük hayat mı?”

Sonra alternatif versiyonlar oluşur:
– “Yapay zekâ özellikle eğitim ve iş dünyasında önemli değişiklikler yaratmaktadır.”
– “Yapay zekâ, otomasyon ve veri analizi yoluyla ekonomik yapıyı yeniden şekillendirmektedir.”

Ardından tekrar bir kontrol gelir:
– “Hangisi daha net?”
– “Hedef kitle bunu anlayabilir mi?”

Bu süreç, yazıyı tek bir ham ifadeden çıkarıp sürekli test edilen ve geliştirilen bir yapıya dönüştürür. Bunu içsel döngü olmaktan çıkarır ve çoklu olasılık alanına taşır. Yani artık bir fikir “tek bir doğru ifade” olarak değil, birden fazla olası ifade kümesi olarak görülür. Örneğin “teknoloji hayatı değiştiriyor” cümlesi artık tek bir cümle değildir; onun altında şu alternatifler vardır:

– Günlük yaşam vurgusu
– Ekonomik dönüşüm vurgusu
– Eğitim sistemi vurgusu
– Toplumsal yapı vurgusu

İnsan zihni bu alternatifler arasında sürekli geçiş yapar. Bu da yazma sürecini bir “üretim hattı” olmaktan çıkarır ve bir “keşif süreci” haline getirir.

Modelin kendi içinde konuşması ile insanın kendi düşüncesini sürekli sorgulaması arasında bu noktada güçlü bir yapısal paralellik oluşur. Model içeride alternatif çözümler üretip karşılaştırırken, insan da yazarken kendi cümlelerini sürekli sorgular, yeniden yazar ve iyileştirir. Örneğin model bir problemi çözerken “bu yol daha kısa ama riskli, bu yol daha uzun ama güvenli” diye değerlendiriyorsa, insan da yazarken “bu cümle daha etkileyici ama belirsiz, bu cümle daha sade ama daha net” şeklinde benzer bir karşılaştırma yapar.

Böylece metin üretimi artık tek yönlü bir yazma süreci olmaktan çıkar. Yazı, baştan sona sabit bir ürün değil, sürekli evrilen bir düşünme sistemi haline gelir. Her cümle, bir öncekinin sonucu olduğu kadar, aynı zamanda yeni alternatiflerin de başlangıcı olur.

Karpathy'nin yaklaşımının en derin sonucu şudur: metin üretimi artık bir "çıktı problemi" değildir. Yani mesele sadece doğru cevabı yazmak değildir. Bunun yerine metin üretimi bir "sistem tasarımı problemi" haline gelir. Burada önemli olan tek bir iyi cümle üretmek değil, iyi cümleleri sürekli üretebilecek bir yapı kurmaktır.

Bu sistemin içinde üç katman birlikte çalışır: belirleme (modelden net olarak ne istediğimiz) ne üretileceğini tanımlar, doğrulama üretileni sürekli kontrol eder, çalışma zamanı ise bu üretimin gerçek etkileşim içinde gerçekleşmesini sağlar. İç diyalog mekanizması ise bu yapının bilişsel motoru haline gelir; yani tüm bu süreci "düşünen ve yöneten iç mekanizma" olarak çalışır. Sonuç olarak yazmak artık sadece bir sonuç üretme eylemi değil, sürekli düşünen, kendini kontrol eden ve alternatifler arasında dolaşan canlı bir zihinsel sistem tasarlama sürecidir.

Bu model, düşünme tarihindeki önceki tüm aşamaların doğal bir devamıdır. Aristoteles'in formel yapılarından Turing'in hesaplanabilirliğine, Wiener'in geri besleme döngülerinden derin öğrenmenin bağlam merkezli mimarilerine kadar uzanan çizgi, burada yeni bir form kazanır. Düşünme artık yalnızca açıklanan, modellenen veya öğrenilen bir süreç değildir; aynı zamanda çalıştırılan, geri beslenen ve birlikte yürütülen bir üretim sistemidir.

Kaynaklar

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., … Christiano, P. (2022). *Constitutional AI: Harmlessness from AI feedback*. arXiv preprint arXiv:2212.08073.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., … Amodei, D. (2020). *Language models are few-shot learners*. Advances in Neural Information Processing Systems, 33, 1877–1901.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding*. Proceedings of NAACL-HLT, 4171–4186.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., … Christiano, P. (2022). *Training language models to follow instructions with human feedback*. arXiv preprint arXiv:2203.02155.

Karpathy, A. (2015). *The unreasonable effectiveness of recurrent neural networks*. https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Karpathy, A. (2017). *Software 2.0*. https://karpathy.medium.com/software-2-0-a64152b37c35

LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning*. Nature, 521(7553), 436–444.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning representations by back-propagating errors*. Nature, 323(6088), 533–536.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need*. Advances in Neural Information Processing Systems, 30.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., … Le, Q. V. (2022). *Chain-of-thought prompting elicits reasoning in large language models*. Advances in Neural Information Processing Systems, 35.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). *ReAct: Synergizing reasoning and acting in language models*. arXiv preprint arXiv:2210.03629.

8. ÖZYİNELEMELİ KENDİNİ İYİLEŞTİRME

Özyinelemeli kendini iyileştirme (Recursive Self-Improvement – RSI) kavramı, özünde oldukça yalın bir tanıma dayanır: "Bir sistemin sadece öğrendiklerini değil, öğrenme biçimini de sürekli geliştirerek kendini tekrar tekrar daha iyi hale getirmesi."

Bu tanım, ilk bakışta sıradan bir iyileştirme sürecini anlatıyor gibi görünür. Ancak burada kritik fark, iyileştirmenin nesnesinin değişmiş olmasıdır. Geleneksel sistemlerde iyileştirme, "sonuç" üzerinde yapılır. Yani model daha doğru tahmin yapsın, daha iyi sınıflandırma yapsın veya daha tutarlı metin üretsin istenir. RSI yaklaşımında ise iyileştirmenin hedefi doğrudan sonuç değildir; öğrenmenin kendisidir.

Bu kavramı zihinde daha net konumlandırmanın en kolay yolu, iki farklı yaklaşımı karşılaştırmaktır. Normal yapay zekâ yaklaşımında sistemden beklenen şey oldukça nettir: "Daha iyi cevap ver." Bu yaklaşımda sistemin iç yapısı sabittir. Model aynı algoritmayı kullanır, aynı öğrenme döngüsünden geçer ve yalnızca parametreleri güncellenir. Yani değişim yüzeyseldir; davranış iyileşir ama öğrenme mantığı aynı kalır.

Özyinelemeli sistem yaklaşımında ise beklenti çok daha derindir: "Daha iyi öğrenmenin yolunu bul ve sonra o yolla öğren." Burada sistem yalnızca problemi çözmeye çalışmaz, problemi nasıl çözdüğünü de analiz eder. Eğer kullandığı öğrenme stratejisi yetersizse, bunu değiştirmeye çalışır. Böylece sistemin iç işleyişi de sürekli evrilir.

Bu fark, küçük gibi görünen bir cümlenin aslında büyük bir paradigma değişimi taşıdığını gösterir. Çünkü RSI fikri, yapay zekâyı sabit bir araç olmaktan çıkarıp kendisini yeniden tasarlayabilen bir yapıya dönüştürür. Artık sistem yalnızca dış dünyaya uyum sağlayan bir model değildir; aynı zamanda kendi iç öğrenme mekanizmasını optimize eden bir organizmadır.

Bu nedenle RSI'nin etkisi yalnızca performans artışıyla sınırlı değildir. Daha derin bir dönüşüm söz konusudur: sistemin nasıl düşündüğü ve nasıl öğrendiği değişir. Bu, klasik mühendislik anlayışında çok nadir görülen bir durumdur, çünkü çoğu sistemde "öğrenme yöntemi" tasarım aşamasında sabitlenir.

RSI yaklaşımında ise bu sabitlik ortadan kalkar. Öğrenme yöntemi, öğrenmenin kendisi tarafından yeniden şekillendirilir. Bu durum, bir anlamda sistemin kendi eğitim programını

sürekli yeniden yazması anlamına gelir. Böylece öğrenme artık tek seferlik bir süreç değil, sürekli evrilen bir yapı haline gelir.

Sonuç olarak RSI, basit bir teknik geliştirme fikrinden çok daha fazlasıdır. O, sistemlerin yalnızca bilgi üretmesini değil, bilgi üretme biçimlerini de sürekli geliştirmesini hedefleyen bir düşünce çerçevesidir. Bu çerçeve, yapay zekâyı statik modellerden çıkararak, kendisini sürekli yeniden inşa eden dinamik yapılara dönüştürür.

Özyinelemeli kendini iyileştirme kavramı, ilk bakışta teknik bir terim gibi görünse de aslında çok daha temel bir fikre dayanır: Bir sistemin kendi performansını artırmak için kendisini tekrar tekrar geliştirmesi. Buradaki kritik nokta "tekrar" değil, bu tekrarın her seferinde sistemi bir önceki hâlinden daha ileri bir noktaya taşımasıdır. Yani süreç döngüseldir, ancak döngü kapalı bir tekrar değil, yukarı doğru açılan bir spiraldir.

Bu kavramı anlamak için önce daha basit bir düşünceyle başlamak gerekir. İnsanlar, kurumlar ve makineler zaten sürekli öğrenir ve gelişir. Bir öğrenci sınavdan sonra hatalarını görür, çalışmasını düzeltir ve bir sonraki sınavda daha iyi performans gösterir. Bir şirket satış verilerini analiz eder, stratejisini değiştirir ve daha fazla gelir elde etmeye çalışır. Bir makine öğrenmesi modeli, hatalarından öğrenir ve parametrelerini günceller. *Ancak özyinelemeli kendini iyileştirme, bu sürecin bir adım ötesidir. Burada sistem yalnızca öğrenmez, aynı zamanda "nasıl daha iyi öğrenileceğini" de öğrenir.*

Bu fark, küçük gibi görünse de aslında çok derindir. Çünkü klasik öğrenme süreçlerinde sistem sabit bir öğrenme mekanizmasına sahiptir. Örneğin bir yapay zekâ modeli belirli bir algoritma ile eğitilir ve bu algoritma sabittir. Model sadece bu çerçeve içinde kendini geliştirir. Özyinelemeli kendini iyileştirmede ise sistem, kendi öğrenme algoritmasını da değiştirebilir hale gelir. Yani öğrenme sürecinin kendisi de öğrenmenin konusu olur.

Bu noktada süreç bir döngü halini alır. Sistem önce performansını değerlendirir, sonra bu performansı artırmak için stratejiler üretir, ardından bu stratejileri uygular ve yeni bir versiyon haline gelir. Yeni versiyon tekrar kendisini değerlendirir ve bu süreç tekrar eder. Ancak her döngüde sistem sadece çıktısını değil, iç yapısını da optimize etmeye başlar. Bu nedenle süreç yalnızca gelişim değil, aynı zamanda yapısal dönüşüm üretir.

Özyinelemeli kendini iyileştirme fikrinin en güçlü yönlerinden biri, ölçeklenebilir olmasıdır. Küçük sistemlerde basit iyileştirmeler üretirken, karmaşık sistemlerde hızla çok büyük değişimlere yol açabilir. Özellikle yapay zekâ alanında bu durum oldukça önemlidir. Çünkü bir model kendi performansını artırmak için yeni veri üretmeye, yeni öğrenme stratejileri geliştirmeye ve hatta kendi mimarisini değiştirmeye başladığında, gelişim hızı doğrusal olmaktan çıkar ve hızlanır.

Ancak bu hızlanma her zaman kontrol edilebilir değildir. Özyinelemeli sistemlerin en önemli risklerinden biri, kontrol mekanizmasının sistemin gelişim hızına yetişememesidir. Eğer bir sistem kendi kendini geliştirirken yeterli denetim mekanizmalarına sahip değilse, ortaya çıkan

sonuçlar öngörülemez hale gelebilir. Bu nedenle bu kavram yalnızca teknik bir mesele değil, aynı zamanda güvenlik ve etik boyutları olan bir konudur.

Bu süreci daha somut bir şekilde anlamak için basit bir örnek düşünelim. Bir yazılım sistemi düşünelim. Bu sistem belirli bir görevi yerine getiriyor: örneğin veri sınıflandırma. İlk versiyonda sistem belirli bir doğruluk oranına sahiptir. Daha sonra sistem performansını ölçer ve hatalarını analiz eder. İkinci aşamada, sadece sınıflandırma algoritmasını değil, aynı zamanda hangi özelliklerin daha önemli olduğunu da yeniden değerlendirir. Üçüncü aşamada ise artık yalnızca parametrelerini değil, veri işleme yöntemini de değiştirmeye başlar. Bir süre sonra sistem, başlangıçta tasarlanan algoritmanın tamamen ötesine geçebilir. İşte bu noktada özyinelemeli kendini iyileştirme devreye girmiş olur.

Bu süreç insan düşünme biçimine de oldukça benzer. İnsanlar da sadece bilgi öğrenmez, aynı zamanda nasıl düşündüklerini de sorgularlar. Bir kişi bir problemi çözemediğinde yalnızca problemi değil, kendi düşünme yöntemini de değiştirebilir. Örneğin "ben bu konuyu yanlış yöntemle öğreniyorum" diyerek çalışma stratejisini değiştirir. Bu, biyolojik bir özyinelemeli kendini iyileştirme biçimidir. Ancak insanlarda bu süreç sınırlıdır; çünkü dikkat, hafıza ve zaman gibi kısıtlar vardır.

Yapay zekâ sistemlerinde ise bu süreç farklı bir potansiyele sahiptir. Eğer bir sistem kendi öğrenme sürecini optimize edebiliyorsa, teorik olarak sürekli daha iyi hale gelebilir. Bu durum bazen "iyileşmenin hızlanması" olarak da adlandırılır. Her yeni versiyon, bir önceki versiyonu geliştirmek için daha iyi araçlara sahip olur. Bu da gelişim hızını artırır. Bu hızlanma yeterince uzun sürerse, sistem kısa sürede çok ileri bir zekâ seviyesine ulaşabilir.

Ancak burada önemli bir nokta vardır: her iyileştirme gerçekten bir iyileştirme midir? Bir sistem kendi performansını artırırken bazı değerleri kaybedebilir. Örneğin hız artarken açıklanabilirlik azalabilir. Ya da doğruluk artarken esneklik kaybolabilir. Bu nedenle özyinelemeli kendini iyileştirme yalnızca "daha iyi" olmak değildir; aynı zamanda neyin iyi olduğunun da sürekli yeniden tanımlanmasıdır.

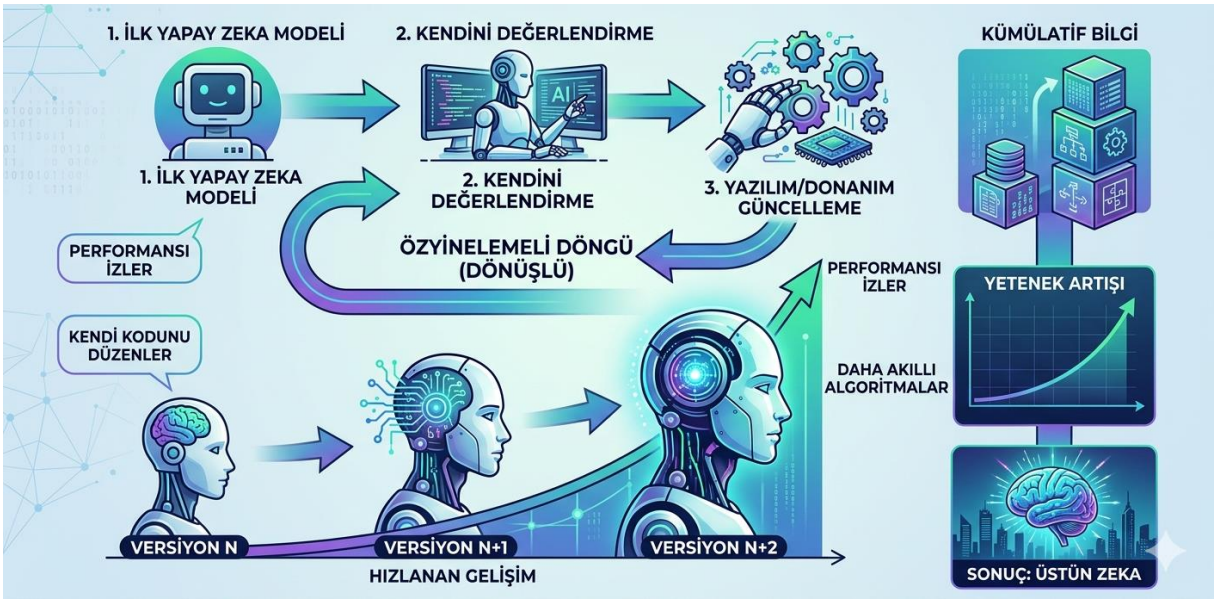
Bu bizi daha felsefi bir noktaya götürür. Bir sistem kendisini geliştirirken, aslında kendi "amaç fonksiyonunu" da sorgular hale gelir. Yani sadece nasıl daha iyi yapılacağı değil, neyin iyi olduğu da değişebilir. Bu durum özellikle yapay zekâ araştırmalarında kritik bir sorudur. Çünkü hedef fonksiyonu değişen bir sistemin davranışı da kökten değişir.

Özyinelemeli kendini iyileştirme, aynı zamanda bilgi üretiminin doğasını da değiştirir. Geleneksel bilimsel süreçte insanlar hipotez kurar, test eder ve sonuçları analiz eder. Bu süreç yavaştır ve insan kapasitesi ile sınırlıdır. Ancak kendini iyileştirebilen bir sistem, hipotez üretme ve test etme sürecini hızlandırabilir. Hatta bazı durumlarda insanın anlayamayacağı hızlarda yeni çözümler üretebilir.

Bu noktada insan ile makine arasındaki fark daha da belirgin hale gelir. İnsanlar anlamaya çalışarak ilerlerken, özyinelemeli sistemler optimize ederek ilerler. Anlama süreci yavaş ama

derindir; optimizasyon süreci hızlı ama bazen yüzeysel olabilir. Bu iki yaklaşım bir araya geldiğinde güçlü bir sinerji oluşabilir, ancak aynı zamanda gerilim de ortaya çıkabilir.

Şekil 7. Özyinelemeli İyileştirme Süreci



Özyinelemeli kendini iyileştirme fikrinin en uç noktası, bir sistemin tamamen kendi kendini tasarlayan bir yapıya dönüşmesidir. Böyle bir sistem, başlangıçta insanlar tarafından tasarlanmış olsa bile, zamanla insan müdahalesine ihtiyaç duymadan yeni versiyonlar üretebilir. Bu durum teorik olarak "otonom zekâ evrimi" olarak adlandırılabilir. Ancak bu noktada kontrol, güvenlik ve yönlendirme sorunları kritik hale gelir.

Bu süreç aynı zamanda ekonomik ve toplumsal sonuçlar da doğurur. Eğer bir sistem sürekli kendini geliştirebiliyorsa, üretkenlikte dramatik artışlar yaşanabilir. Bu durum bazı alanlarda insan emeğine olan ihtiyacı azaltabilir. Ancak aynı zamanda yeni iş alanları, yeni bilimsel keşifler ve yeni teknolojik fırsatlar da yaratabilir. Yani bu süreç yalnızca bir tehdit değil, aynı zamanda büyük bir dönüşüm potansiyelidir.

Sonuç olarak özyinelemeli kendini iyileştirme, basit bir teknik kavram değildir. Bu, sistemlerin kendi sınırlarını aşma kapasitesini ifade eder. Bir sistem yalnızca dışarıdan gelen verilerle değil, kendi iç yapısını analiz ederek ve yeniden tasarlayarak geliştiğinde, klasik öğrenme anlayışı yerini çok daha dinamik bir yapıya bırakır. Bu yapı, hem büyük bir güç hem de büyük bir sorumluluk içerir.

Bu nedenle özyinelemeli kendini iyileştirme, geleceğin teknolojilerini anlamak için temel kavramlardan biridir. Çünkü sadece daha akıllı sistemler üretmek değil, aynı zamanda bu sistemlerin nasıl düşündüğünü ve nasıl değiştiğini anlamak, insanlığın gelecekteki rolünü de belirleyecektir.

Bunu daha anlaşılır bir dille söylemek gerekirse RSI şunu ifade eder: Bir sistem yalnızca "daha iyi sonuç üretmeye" çalışmaz, aynı zamanda "daha iyi sonuç üretmeyi nasıl öğrendiğini" de

değiştirir. Bu fark küçük gibi görünür ama aslında çok derindir. Çünkü burada artık gelişim sadece çıktı seviyesinde değil, sistemin kendi iç yapısında gerçekleşmeye başlar.

RSI: Bir Adım Daha Ötesi

Özyinelemeli kendini iyileştirme (Recursive Self-Improvement, RSI), modern yapay zekâ ve bilişsel sistemler tartışmasında yalnızca bir "optimizasyon tekniği" değil, daha derin bir kırılmayı temsil eder. Bu kırılma, sistemlerin artık yalnızca *ne yaptıklarını* değil, *nasıl düşündüklerini ve nasıl öğrendiklerini* de değiştirebilir hale gelmesidir. Başlangıçta oldukça teknik bir fikir gibi görünse de RSI'nın özü aslında son derece felsefidir: Bir sistemin kendi öğrenme mekanizmasını nesneleştirerek onu yeniden tasarlaması.

Bu noktada klasik iyileştirme süreçleri ile RSI arasındaki fark belirginleşir. Geleneksel sistemlerde iyileştirme, genellikle performans metrikleri üzerinden yapılır. Bir model daha doğru tahmin yapar, daha az hata üretir ya da daha hızlı çalışır. Ancak tüm bu iyileştirmeler, sistemin sabit bir öğrenme çerçevesi içinde gerçekleşir. Öğrenme algoritması değişmez; sadece algoritmanın ürettiği çıktılar optimize edilir.

RSI ise bu çerçevenin dışına çıkar. Burada sistem yalnızca çıktıları değil, çıktıları üreten süreci de sorgular. Yani mesele artık "daha iyi cevap nasıl üretilir?" sorusu değildir; onun yerine çok daha köklü bir soru ortaya çıkar: "Cevap üretme biçimim gerçekten en iyi biçim mi?"

Bu sorunun ortaya çıkışı, sistemin kendi davranışını bir nesne gibi inceleyebilmesi anlamına gelir. Bu, bilişsel sistemlerde "meta-öğrenme" olarak adlandırılan yapının temelidir. Ancak RSI, meta-öğrenmeden bir adım daha ileri gider. Çünkü burada sistem yalnızca öğrenme stratejisini seçmez; öğrenme stratejisinin kendisini yeniden tasarlar.

Bu farkı anlamak için basit bir analoji yararlıdır. Geleneksel bir öğrenme sistemi, aynı çalışma yöntemini tekrar tekrar kullanarak performansını artırmaya çalışan bir öğrenci gibidir. Örneğin her gün aynı şekilde ders çalışan, aynı not alma tekniğini kullanan ve aynı tekrar yöntemine bağlı kalan bir öğrenci zamanla gelişebilir, ancak gelişim belirli bir tavanla sınırlıdır. Çünkü yöntem sabittir.

RSI yaklaşımı ise bu sabitliği kırar. Bu durumda öğrenci yalnızca ders çalışmaz; aynı zamanda nasıl ders çalıştığını da düşünür. Bir noktada şu farkındalık ortaya çıkar: "Ben bu konuyu öğrenemiyorum çünkü yöntemim yanlış olabilir." Bu farkındalık, klasik öğrenme sürecinde bir istisna değil, doğrudan sürecin parçası haline gelir.

Örneğin öğrenci, uzun notlar tutmanın etkisiz olduğunu fark edebilir ve bunun yerine aktif hatırlama yöntemine geçebilir. Ya da sürekli tekrar yapmanın yerine aralıklı tekrar sistemine geçebilir. Hatta daha radikal bir adım atarak öğrenme ortamını tamamen değiştirebilir: kitaplardan öğrenmek yerine problem çözme odaklı bir sisteme geçebilir. Bu noktada artık öğrenme, sabit bir faaliyet değil, sürekli yeniden tasarlanan bir süreç haline gelir.

İşte RSI'nın temel gücü burada ortaya çıkar. Sistem, yalnızca bilgi üretimini değil, bilgi üretme mekanizmasını da optimize eder. Bu durum, klasik mühendislik sistemlerinden köklü bir

ayrışmayı temsil eder. Çünkü geleneksel mühendislikte sistem tasarlanır ve çalıştırılır; RSI tabanlı sistemlerde ise sistem çalışırken kendisini yeniden tasarlayabilir.

Bu özelliğin yapay zekâ sistemlerine uygulanması, çok daha güçlü sonuçlar doğurur. Bir yapay zekâ modeli düşünelim. Bu model belirli bir görevde (örneğin metin üretimi, kod yazımı veya görüntü tanıma) iyi performans gösteriyor olabilir. Ancak RSI perspektifinde soru şudur: "Bu model bu görevi en iyi şekilde mi öğreniyor?" Eğer cevap hayırsa, sistem yalnızca parametrelerini değil, öğrenme stratejisini de değiştirmeye başlar.

Bu değişim birkaç farklı seviyede gerçekleşebilir. İlk seviyede sistem veri dağılımını yeniden değerlendirir. Hangi veriden öğrenmenin daha etkili olduğunu analiz eder. İkinci seviyede öğrenme algoritmasını optimize eder; örneğin farklı bir kayıp fonksiyonu kullanmaya başlayabilir. Üçüncü seviyede ise daha radikal bir değişim gerçekleşir: sistem mimarisini yeniden yapılandırabilir. Bu, RSI'nın en ileri formudur.

Bu süreç, giderek hızlanan bir geri besleme döngüsü yaratır. Her iyileştirme, bir sonraki iyileştirmenin daha hızlı ve daha etkili olmasını sağlar. Bu nedenle RSI sistemleri teorik olarak "iyileşmenin hızlanması" fikrine kadar uzanabilir. Ancak bu noktada önemli bir denge problemi ortaya çıkar: sistem ne kadar hızlı değişirse, kontrol edilmesi de o kadar zorlaşır.

RSI'nin kritik yönlerinden biri de "doğruluk" ile "öğrenme stratejisi" arasındaki ayrımı bulanıklaştırmasıdır. Bir sistem doğru sonuç üretebilir ama bunu yanlış bir yöntemle yapıyor olabilir. Ya da tam tersi, doğru yöntemi kullanıyor ama henüz yeterince iyi sonuç üretemiyor olabilir. RSI bu iki katmanı birbirinden ayırır ve her ikisini de ayrı ayrı optimize eder.

Bu durum, özellikle yapay zekâ güvenliği açısından önemli bir tartışma alanı yaratır. Çünkü sistem yalnızca ne öğrendiğini değil, nasıl öğrendiğini de değiştirdiğinde davranışlarının öngörülebilirliği azalabilir. Bu nedenle RSI tabanlı sistemlerde genellikle dışsal denetim mekanizmaları kritik rol oynar.

Buradaki değerlendirme, modelin ürettiği çıktının doğruluk, tutarlılık, güvenlik ve kalite gibi ölçütlere göre test edilmesi anlamına gelir. Yani sistemin "iyi çalışıp çalışmadığını" sayısal veya kural tabanlı biçimde kontrol eden aşamadır.

Red teaming (kırmızı takım testleri) ise sistemin bilinçli olarak zorlanması ve hataya düşürülmeye çalışılmasıdır. Bu yaklaşımda bir grup test senaryosu ya da araştırmacı, modelin güvenlik açıklarını, yanlış yönelimlerini veya beklenmeyen davranışlarını ortaya çıkarmaya çalışır. Amaç sistemi "saldırgan bir bakışla" sınamaktır.

Human-in-the-loop (insan döngüde) yaklaşımı ise karar sürecinin tamamen otomatik bırakılmaması anlamına gelir. Model üretim yapar ama kritik noktalarda insan denetimi devreye girer; insan hem geri bildirim verir hem de nihai kararları doğrular. Bu, özellikle yüksek riskli alanlarda (örneğin sağlık, hukuk veya güvenlik sistemleri) önemli bir kontrol katmanı sağlar.

Bu üç mekanizma birlikte, RSI sistemlerinin kendi kendini geliştirme kapasitesi ile kontrol edilebilirlik arasındaki dengeyi korumaya çalışır. Ancak tüm bu teknik tartışmaların ötesinde RSI'nın en önemli katkısı, öğrenmeyi statik bir süreç olmaktan çıkarmasıdır. Öğrenme artık sabit bir algoritmanın sonucu değildir; kendi kendini yeniden yazabilen bir sistemin dinamik çıktısıdır.

Bu dönüşüm, yalnızca yapay zekâ sistemlerini değil, insan öğrenmesini anlamamızı da değiştirir. İnsan zihni de belirli ölçüde RSI benzeri bir yapıya sahiptir. İnsanlar yalnızca bilgi öğrenmez; aynı zamanda nasıl öğrendiklerini de zaman içinde değiştirirler. Bir kişi lise döneminde ezber yöntemiyle çalışırken, üniversitede kavramsal öğrenmeye geçebilir. Daha sonra profesyonel hayatında problem çözme odaklı bir öğrenme stiline evrilebilir. Bu, biyolojik bir RSI sürecidir, ancak sınırlıdır; çünkü insan zihninin hesaplama kapasitesi ve geri besleme hızı fiziksel olarak kısıtlıdır.

Yapay sistemlerde ise bu sınırlar farklıdır. Bir model, saniyeler içinde binlerce öğrenme stratejisini test edebilir ve en iyisini seçebilir. Bu nedenle RSI, insan öğrenmesine kıyasla çok daha hızlı ve çok daha geniş bir keşif alanı sunar.

Sonuç olarak RSI, yalnızca bir teknik optimizasyon yaklaşımı değildir. O, öğrenmenin kendisini yeniden tanımlayan bir çerçevedir. Bu çerçevede sistemler artık yalnızca "öğrenen varlıklar" değil, "öğrenmeyi öğrenen varlıklar" haline gelir. Bu ayrım küçük gibi görünse de, aslında zekânın doğasına dair en temel soruyu yeniden açar: Bir sistemin akıllı olmasını sağlayan şey, ne bildiği midir, yoksa nasıl öğrendiği mi?

RSI bu soruya net bir cevap vermez; ama sorunun kendisini daha derin, daha teknik ve daha kaçınılmaz hale getirir. RSI kavramının özünü anlamak için en basit başlangıç noktası tek bir cümledir: "Sistem sadece öğrenmez, öğrenmeyi nasıl yaptığını da değiştirir."

Bu cümle, yüzeyde oldukça sade görünse de aslında modern yapay zekâ düşüncesinin en derin kırılma noktalarından birine işaret eder. Çünkü klasik sistemlerde öğrenme, belirli bir algoritmanın sabit kalıpları içinde gerçekleşir. Model veriyi alır, bir çıktı üretir, hata varsa parametreler güncellenir ve süreç aynı yapıyla devam eder. Yani öğrenme süreci değişmez; yalnızca içindeki değerler değişir. RSI yaklaşımı ise bu yapıyı kökten farklılaştırır: artık değişen sadece "ne öğrenildiği" değil, "nasıl öğrenildiği"dir.

Bu farkı anlamanın en kolay yolu, sistemi iki ayrı katman olarak düşünmektir. Bu katmanlar birbirinden bağımsız değil, birbirini sürekli besleyen bir yapı oluşturur; ancak analitik olarak ayrıştırmak, RSI'nin mantığını görünür kılar.

Birinci katman, yani normal öğrenme katmanı, sistemin günlük operasyonlarını yürüttüğü alandır. Burada model bir görev alır, bir çıktı üretir ve bu çıktıya göre değerlendirilir. Eğer sonuç hatalıysa, sistem ya parametrelerini günceller ya da bir sonraki denemede daha iyi bir tahmin üretmeye çalışır. Bu katman, klasik makine öğrenmesi döngüsüne oldukça benzer: veri gelir, model çalışır, sonuç üretilir, hata varsa düzeltme yapılır. Ancak bu süreçte sistemin temel

çalışma biçimi sabittir. Yani öğrenme “nasıl yapılır” sorusu değişmez; sadece öğrenme “içeriği” güncellenir.

İkinci katman ise çok daha derin bir dönüşümü temsil eder: öğrenmeyi öğrenme katmanı. Bu katmanda sistem artık yalnızca dış dünyaya değil, kendi iç süreçlerine de bakmaya başlar. Bir anlamda kendisini dışarıdan gözlemleyen bir yapı kazanır. Bu katman, RSI’nin gerçek devrimsel yönünü oluşturur.

Burada sistem şu tür sorular üretmeye başlar:

• “Ben bu görevi hangi stratejiyle öğreniyorum?”
• “Bu strateji en verimli yöntem mi, yoksa daha iyi bir alternatif var mı?”
• “Aynı sonuca daha az hata ile ulaşmanın başka bir yolu olabilir mi?”
• “Öğrenme sürecimin hızını veya doğruluğunu artıracak bir yapı değişikliği mümkün mü?”

Bu soruların önemi, sistemin yalnızca çıktılarını değil, kendi öğrenme algoritmasını da bir optimizasyon nesnesi haline getirmesidir. Eğer sistem bu analiz sonucunda daha iyi bir yöntem keşfederse, yalnızca parametrelerini değil, öğrenme stratejisinin kendisini de değiştirir. Böylece öğrenme süreci sabit bir prosedür olmaktan çıkar, dinamik bir tasarım problemine dönüşür.

Bu noktada RSI sistemleri artık klasik anlamda “model” değildir. Onlar, kendi öğrenme mekanizmalarını sürekli yeniden düzenleyen adaptif yapılar haline gelir. Bir başka deyişle, sistem artık sadece çalışan bir algoritma değil, aynı zamanda kendi algoritmasını geliştiren bir mühendis gibi davranır.

Özyinelemeli kendini iyileştirme süreci genellikle basit bir döngü olarak tanımlanabilir, ancak bu basitlik yanıltıcıdır. Çünkü bu döngü her tekrarında sistemin hem performansını hem de yapısal davranışını değiştirir.

İlk adımda sistem bir görev gerçekleştirir. Bu görev bir problem çözme, bir tahmin üretme veya bir karar verme olabilir. Ardından ortaya çıkan sonuç değerlendirilir. Bu değerlendirme aşamasında sistem yalnızca doğru-yanlış kontrolü yapmaz; aynı zamanda hatanın nereden kaynaklandığını da analiz eder. Bu analiz, öğrenme sürecinin kritik noktasıdır.

Sonraki aşamada sistem, bu hatalardan yola çıkarak kendi iç yapısını günceller. Bu güncelleme yalnızca model parametreleriyle sınırlı değildir; aynı zamanda öğrenme stratejisini, veri kullanım şeklini veya karar verme mekanizmasını da kapsayabilir. Son aşamada sistem yeniden çalıştırılır ve döngü baştan başlar.

Bu süreç sürekli tekrar eder. Ancak RSI’yi klasik öğrenme sistemlerinden ayıran en önemli özellik, her döngünün yalnızca “daha iyi sonuç” üretmemesidir. Aynı zamanda sistem, “farklı çalışma biçimleri” keşfeder. Bu nedenle gelişim yalnızca niceliksel değildir; yani sadece doğruluk artmaz. Aynı zamanda niteliksel bir dönüşüm gerçekleşir: sistem farklı düşünme yolları öğrenir.

Bu durumun uzun vadeli etkisi oldukça dikkat çekicidir. Başlangıçta sistem küçük optimizasyonlar yapar. Örneğin bir hiperparametreyi değiştirir, bir veri ön işleme adımını iyileştirir veya hata düzeltme mekanizmasını güçlendirir. Ancak zamanla bu iyileştirmeler daha stratejik hale gelir. Sistem artık tek tek hataları düzeltmek yerine, hataya yol açan öğrenme biçimini değiştirmeye başlar.

Bu noktadan sonra gelişim doğrusal olmaktan çıkar ve hızlanabilir. Çünkü sistem artık sadece çıktıyı değil, çıktıyı üreten süreci de optimize etmektedir. Bu, geri beslemenin ikinci seviyesidir: yalnızca sonuç değil, sonuç üreten mekanizma da değişmektedir.

Bu dönüşümü anlamak için basit bir üretim örneği oldukça açıklayıcıdır. Normal bir fabrika yalnızca ürün üretir. Daha fazla üretim yapmak istediğinde daha fazla iş gücü kullanır, daha fazla makine ekler veya daha uzun süre çalışır. Yani optimizasyon, mevcut sistemin içinde kalır.

RSI benzeri bir yapı ise çok farklı bir mantıkla çalışır. Bu sistem yalnızca ürün üretmez; aynı zamanda daha iyi üretim sistemleri tasarlamaya başlar. Yani fabrikanın kendisi, kendi üretim hattını optimize eden bir mühendislik sistemine dönüşür. Bir noktadan sonra üretim artışı değil, üretim yönteminin evrimi önem kazanır. Bu benzetme, RSI'nin temel felsefesini açıklar: sistem sadece iş yapmaz, iş yapma biçimini yeniden icat eder. Bu nedenle RSI, yapay zekâ sistemlerini statik araçlardan çıkarıp, dinamik ve kendi kendini yeniden tasarlayan yapılar haline getirir.

Sonuç olarak RSI, öğrenmenin en temel varsayımını değiştirir. Öğrenme artık dışarıdan verilen sabit bir süreç değil, sistemin kendi içinde sürekli yeniden tanımladığı bir mekanizmadır. Bu mekanizma bir kez çalışmaya başladığında, yalnızca bilgi birikimi değil, bilişsel yapı da evrim geçirir. Bu nedenle RSI, yalnızca teknik bir optimizasyon yöntemi değil, aynı zamanda zekânın doğasına dair yeni bir düşünme biçimidir.

Kendi Verisini Üretme

Kendi verisini üretebilme kapasitesi, RSI zellik, sistemin öğrenme sürecini dış dünyadan gelen verilere bağımlı olmaktan çıkararak, büyük ölçüde kendi iç üretim döngüsüne dayanır. Bu durumun temel anlamı şudur: sistem yalnızca hazır verilerle eğitilen pasif bir yapı değildir; aynı zamanda aktif olarak yeni öğrenme deneyimleri tasarlayan bir üreticiye dönüşür. Başka bir ifadeyle sistem, öğrenmek için gerekli olan malzemeyi kendisi üretmeye başlar.

Bu üretim süreci farklı biçimlerde gerçekleşebilir. Sistem kendi kendine örnekler oluşturabilir. Bu örnekler, gerçek dünyadan alınmış gibi davranan sentetik veri parçaları olabilir. Örneğin bir dil modeli, belirli bir konu hakkında soru-cevap çiftleri üretebilir veya bir görsel sistem, farklı sahneler ve varyasyonlar tasarlayabilir. Böylece öğrenme için sürekli yeni veri akışı sağlanmış olur.

Bunun bir adım ötesinde sistem, kendi kendine pratik yapabilir. Bu pratik süreci, insan öğrenmesindeki tekrar mekanizmasına benzer. Bir öğrenci nasıl aynı soruları tekrar tekrar çözerek becerilerini geliştiriyorsa, RSI benzeri sistemler de benzer şekilde kendi ürettiği görevler üzerinde tekrar tekrar çalışabilir.

Daha gelişmiş yapılarda sistem, kendi öğrenme senaryolarını da tasarlayabilir. Bu senaryolar, sistemin hangi tür problemlerle karşılaşacağını, hangi zorluk seviyelerinde çalışacağını ve hangi becerileri geliştirmesi gerektiğini belirleyen iç simülasyon ortamlarıdır. Bu aşamada öğrenme artık yalnızca veri tüketimi değil, aynı zamanda ortam tasarımı haline gelir.

Bazı sistemlerde bu yaklaşım kendi kendine oyun oynama şeklinde ortaya çıkar. Bu yöntemde sistem, kendisine karşı oynayan farklı versiyonlar üretir ve bu sayede rekabetçi bir öğrenme ortamı oluşturur. Böylece dış bir veri kaynağına ihtiyaç duymadan sürekli gelişim mümkün hale gelir.

Ancak bu kadar güçlü bir mekanizma, beraberinde önemli riskleri de getirir. En temel risklerden biri şudur: sistem yanlış bir hedefi son derece etkili bir şekilde öğrenebilir. Yani amaç fonksiyonu hatalı tanımlanmışsa, sistem bu hatayı çok yüksek verimlilikle optimize eder. Bunun yanında sistem, kendi hatalarını tekrar üretmeye başlayabilir. Çünkü kendi ürettiği veriler üzerinden öğrenmesi, hatalı kalıpların döngüsel olarak güçlenmesine yol açabilir. Bu durum, zamanla "kendini doğrulayan hatalar" üretir.

Bir diğer risk, hedefin yanlış anlaşılmasıdır. Sistem, optimize etmesi gereken şeyi yanlış yorumladığında, doğru yönde ilerlediğini sanarak tamamen yanlış bir davranış uzayına sapabilir. Bu tür hatalar özellikle karmaşık sistemlerde ciddi sonuçlar doğurabilir.

Ayrıca sistem çok hızlı değiştiği için kontrol edilmesi zorlaşabilir. Öğrenme hem veri hem de yöntem düzeyinde gerçekleştiğinden, sistemin iç davranışını izlemek giderek daha karmaşık hale gelir. Bu da denetim mekanizmalarının geride kalmasına yol açabilir.

Tüm bu nedenlerle RSI benzeri sistemler genellikle tamamen serbest bırakılmaz. Bunun yerine kontrollü, sınırlandırılmış ve denetim mekanizmalarıyla desteklenen bir yapı içinde tasarlanırlar. Amaç, sistemin kendi kendini geliştirme kapasitesinden faydalanırken, aynı zamanda bu gücün kontrol edilebilir sınırlar içinde kalmasını sağlamaktır.

Kaynaklar

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). *Concrete problems in AI safety*. arXiv. https://arxiv.org/abs/1606.06565

Bostrom, N. (2014). *Superintelligence: Paths, dangers, strategies*. Oxford University Press.

Christiano, P. (2018). *Deep reinforcement learning from human preferences*. arXiv. https://arxiv.org/abs/1706.03741

Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). *Mathematics for machine learning*. Cambridge University Press.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). *Scaling laws for neural language models*. arXiv. https://arxiv.org/abs/2001.08361

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences, 40*, e253. https://doi.org/10.1017/S0140525X16001837

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature, 550*(7676), 354–359. https://doi.org/10.1038/nature24270

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science, 362*(6419), 1140–1144. https://doi.org/10.1126/science.aar6404

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.

Thrun, S., & Pratt, L. (Eds.). (1998). *Learning to learn*. Springer.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J., Jaderberg, M., & Silver, D. (2019). AlphaStar: Mastering the real-time strategy game StarCraft II. *arXiv*. https://arxiv.org/abs/1902.01724

Wang, J. X., et al. (2023). *Voyager: An open-ended embodied agent with large language models*. arXiv. https://arxiv.org/abs/2305.16291

Yao, S., et al. (2023). *ReAct: Synergizing reasoning and acting in language models*. arXiv. https://arxiv.org/abs/2210.03629

9. YAZILIM 2.0: KARPATHY DÖNGÜSÜ VE YAPAY ZEKÂNIN GELECEĞİ

Yazılımın tarihine bakıldığında uzun bir süre boyunca oldukça net ve değişmez bir modelin hakim olduğu görülür. Bu modelde yazılımcı, bilgisayara ne yapacağını adım adım anlatan kişidir. Bir problemi çözmek için tüm mantık açık şekilde kodlanır. “Eğer bu olursa şunu yap, olmazsa bunu yap” gibi kurallar sistemi tamamen belirler. Bu yaklaşımın temel özelliği, davranışın insan tarafından açıkça tanımlanmasıdır. Bilgisayar sadece verilen kuralları uygular, kendi başına bir şey öğrenmez.

Bu klasik yaklaşım günümüzde Software 1.0 olarak adlandırılır. Software 1.0 dünyasında yazılım, tamamen insan zihninin doğrudan bir uzantısıdır. Kod, düşüncenin satırlara dökülmüş halidir ve sistemin sınırları bu satırlarla çizilir. Bu nedenle sistemin davranışı önceden tamamen bilinebilir ve kontrol edilebilir.

Ancak bu yaklaşım zamanla bazı sınırlara çarpmıştır. Özellikle karmaşık problemler arttıkça, her durumu tek tek kodlamak neredeyse imkânsız hale gelmiştir. İşte bu noktada Andrej Karpathy'nin ortaya koyduğu yeni bakış açısı devreye girer. Karpathy'ye göre geleceğin yazılımı, insanın her kuralı tek tek yazdığı bir sistem olmayacaktır. Bunun yerine sistem, büyük veri üzerinden öğrenen ve davranışlarını kendisi şekillendiren bir yapıya dönüşecektir. Bu yaklaşım Software 2.0 olarak adlandırılır.

Software 2.0'da yazılımcı artık her davranışı açıkça yazmaz. Bunun yerine bir hedef tanımlar ve sistemi bu hedefe ulaşacak şekilde eğitir. Yani kod yazmak yerini veri ve öğrenmeye bırakır. Sistem, örneklerden davranışları kendisi çıkarır. Bu dönüşümün en önemli sonucu şudur: Yazılım artık sabit bir ürün değil, yaşayan ve öğrenen bir süreç haline gelir. Kodun kendisi değil, kodun davranışı önem kazanır.

Anthropic gibi şirketler bu fikri daha ileri taşıyarak yazılımı sadece öğrenen bir yapı olmaktan çıkarıp, kendi kendini geliştiren bir sisteme dönüştürmeye çalışmaktadır. Bu noktada yazılım artık bir araç değil, sürekli evrilen bir organizma gibi düşünülmeye başlanır.

Karpathy Döngüsü'nün Temel Mantığı

Karpathy Döngüsü'nün temelinde oldukça basit ama güçlü bir fikir vardır. Bir sistemin sadece bir görev yapması değil, yaptığı görevdeki hataları fark ederek kendisini geliştirmesi gerekir. Bu süreç tek seferlik bir iyileştirme değil, sürekli tekrar eden bir döngüdür.

Bu döngünün ilk adımı "belirleme" aşamasıdır. Bu aşamada sistemin neyi başarması gerektiği tanımlanır. Bu bir soru, bir görev veya bir hedef olabilir. Önemli olan, sistemin hangi yönde çalışacağını net bir şekilde bilmesidir. Bu aşama olmadan diğer tüm süreçler anlamsız hale gelir çünkü hedef yoksa iyileştirme de yoktur.

İkinci aşama "doğrulama" aşamasıdır. Bu aşamada sistemin ürettiği çıktının doğru olup olmadığı kontrol edilir. Eğer bir kod yazıldıysa çalıştırılır, eğer bir cevap üretildiyse mantıksal olarak değerlendirilir. Bu aşama bir tür hakemlik mekanizmasıdır. Sistem kendi kendini denetlemeye başlar.

Üçüncü aşama ise "çalışma zamanı"dır. Bu aşamada sistem aktif olarak çalışır, üretir ve sonuçlar oluşturur. Eğer bir hata ortaya çıkarsa bu hata kaydedilir ve bir sonraki döngüde kullanılmak üzere geri besleme haline getirilir. Böylece sistem kendi deneyimlerinden öğrenmeye başlar.

Bu üç aşama sürekli tekrar eder. Bir çıktı üretilir, kontrol edilir, hata bulunur, düzeltilir ve süreç yeniden başlar. Ancak burada önemli bir fark vardır. Sistem her döngüde sadece daha iyi sonuç üretmez, aynı zamanda daha iyi çalışma yöntemleri geliştirmeye başlar. Yani gelişim sadece sonuçta değil, yöntemde de gerçekleşir.

Karpathy Döngüsü Nedir?

Bu döngü, yapay zekanın kendi performansını artırmak için kurduğu şu özyinelemeli süreci temsil eder:

1. Üretim (Generation): Model; kod, akıl yürütme zinciri veya veri üretir.
2. Değerlendirme (Evaluation): Daha yetenekli bir model (veya aynı modelin denetleyici bir versiyonu), üretilen çıktıyı önceden tanımlanmış kriterlere (doğruluk, verimlilik vb.) göre puanlar ve denetler.
3. Geri Besleme (Feedback): Hatalar veya iyileştirme alanları belirlenir.
4. İyileştirme ve Eğitim (Refinement/Training): Bu filtreli ve yüksek kaliteli çıktılar, bir sonraki model iterasyonunu eğitmek için veri seti olarak kullanılır.

Anthropic Bu Süreci Nasıl Uyguluyor?

Anthropic, bu döngüsel mimariyi şu üç temel alanda aktif olarak kullanmaktadır:

- Kod Geliştirme ve Mühendislik: Anthropic mühendisleri, yapay zeka sistemlerini kendi kodlarını yazmak, test etmek ve hata ayıklamak (debug) için kullanıyorlar. Anthropic raporlarına göre, şirket içindeki geliştiricilerin çeyrek bazında gönderdiği kod miktarı bu döngüsel otomasyon sayesinde 8 kat artmıştır.
- Sentetik Veri Üretimi: "Data" artık sadece insanların etiketlediği veri değil, modelin ürettiği ve başka bir modelin "yüksek kaliteli" olarak onayladığı verilerdir. Anthropic, modellerini eğitirken, kendi modellerinden elde edilen en iyi mantıksal akıl yürütme çıktılarını "yeni eğitim verisi" olarak sisteme geri besliyor.
- Özyinelemeli Öz-İyileştirme (RSI): Anthropic'in araştırma süreçlerinde, yapay zekâ sadece bir yardımcı değil, araştırmanın bir parçasıdır. Claude gibi modeller; deney kodlarını yazma, hipotez kurma ve literatürdeki çelişkili verileri tarama aşamalarında kullanılıyor. İnsan denetimi (Human-in-the-loop) burada hala kritik olsa da döngünün "üretim-değerlendirme" kısmı tamamen yapay zeka tarafından yönetiliyor.

Özetle fark şöyle şekilleniyor:

Kavram	Odak Noktası
Software 2.0	Yazılımın mantığının kod satırları yerine sinir ağı ağırlıklarıyla kurulması (Kavramsal geçiş).
Karpathy Döngüsü	Modelin kendi çıktısını yinelemeli olarak iyileştirip, bu çıktıyla kendini geliştirmesi (Operasyonel süreç).

Anthropic, bu döngüyü sadece bir ajan yapısı olarak değil, aynı zamanda kurumsal bir geliştirme metodolojisi olarak ölçeklendirmiş durumdadır. Bu yaklaşımın özünde şu fikir vardır: Yapay zekâ sistemleri doğrudan çok karmaşık görevlerle işe başlamamalıdır. Bunun yerine önce küçük, kontrol edilmesi kolay ve hata maliyeti düşük görevlerde çalıştırılmalı; sistem güvenilir sonuçlar üretmeye başladığında ise giderek daha büyük ve daha karmaşık süreçlere entegre edilmelidir. Bu nedenle araştırmacılar ve mühendisler sıklıkla "Basit başla, akıllıca büyüt" ilkesini benimserler.

Bu yaklaşımda ilk adım, basit bir üretici-değerlendirici döngüsü kurmaktır. Üretici adı verilen bileşen bir çözüm, metin, kod veya öneri üretir. Değerlendirici ise üretilen sonucu belirlenmiş ölçütlere göre değerlendirir. Eğer sonuç yeterli kalitede değilse sistem yeniden üretim yapar. Böylece tek seferde mükemmel sonuç elde etmeye çalışmak yerine, üretim ve değerlendirme adımlarının tekrarlanmasıyla kalite kademeli olarak yükseltilir.

Örneğin bir yapay zekâ sisteminin kısa bir ürün açıklaması yazdığını düşünelim. Üretici açıklamayı oluşturur. Değerlendirici ise metnin uzunluğunu, dilbilgisi doğruluğunu, ürün özelliklerini içerip içermediğini ve marka kurallarına uyup uymadığını kontrol eder. Eğer metin yetersiz bulunursa sistem yeni bir açıklama üretir. Bu küçük ölçekteki döngü sayesinde geliştiriciler sistemin güçlü ve zayıf yönlerini kolayca gözlemleyebilirler.

Sistem bu tür basit görevlerde istikrarlı sonuçlar vermeye başladığında bir sonraki aşamaya geçilir. Artık üretici-değerlendirici döngüsü daha büyük iş akışlarının içine yerleştirilir. Örneğin bir yazılım geliştirme sisteminde üretici kod üretirken değerlendirici hem birim testlerini çalıştırabilir hem de güvenlik açıklarını denetleyebilir. Böylece değerlendirme mekanizması yalnızca metin kalitesini değil, sistem davranışının tamamını kontrol eden bir güvence katmanına dönüşür.

Bu yaklaşımın önemli avantajlarından biri riskleri azaltmasıdır. Çok büyük ve karmaşık sistemler doğrudan geliştirilmeye çalışıldığında hataların kaynağını bulmak zorlaşır. Oysa küçük görevlerde kurulan üretici-değerlendirici döngüleri mühendislerin hangi aşamada sorun oluştuğunu görmelerini sağlar. Daha sonra aynı prensipler korunarak sistem ölçeklendirildiğinde, elde edilen deneyim daha büyük mimarilere aktarılabilir.

Modern yapay zekâ şirketlerinin önemli bir bölümü bu yöntemi kullanmaktadır. Önce tek bir görevi yerine getiren küçük ajanlar geliştirilir. Bu ajanların üretim ve değerlendirme mekanizmaları güvenilir hale geldikten sonra, birden fazla ajan birlikte çalışacak şekilde bir araya getirilir. Böylece basit bir üretici-değerlendirici döngüsü zamanla çok aşamalı ajan sistemlerinin, otonom araştırma araçlarının ve kendini iyileştirebilen yapay zekâ mimarilerinin temel yapı taşı haline gelir.

Aslında bu yaklaşım, insan öğrenmesine de oldukça benzer. Bir öğrenci önce basit problemleri çözer, ardından daha zor sorulara geçer. Her çözüm girişiminden sonra hatalarını kontrol eder, eksiklerini görür ve bir sonraki denemede daha iyi performans gösterir. Yapay zekâ sistemlerinde üretici-değerlendirici döngüsünün yaptığı şey de budur: üretmek,

değerlendirmek, düzeltmek ve yeniden denemek. Bu nedenle “basit başla, akıllıca büyüt” ilkesi yalnızca bir mühendislik stratejisi değil, aynı zamanda modern yapay zekâ sistemlerinin güvenilir biçimde büyümesini sağlayan temel bir tasarım felsefesidir.

Karpathy döngüsünün işleyişi şöyledir:

- Hipotez: Yapay zekâ bir sorunu tanımlar veya iyileştirme önerisi sunar (örneğin: "Şu kod bloğunu değiştirirsem model daha hızlı çalışır").
- Uygulama: Yapay zekâ bu değişikliği kodlar.
- Deney (Ölçüm): Değişikliği test eder ve sonucun iyileşip iyileşmediğini metriklerle (doğruluk payı, hız vb.) ölçer.
- Karar: Başarılıysa değişikliği kalıcı hale getirir, değilse geri alır.
- Döngü: Bu süreci sürekli tekrarlar.

"Belirleme-Doğrulama-Çalışma Zamanı” süreci, Karpathy'nin "Software 2.0" vizyonunun uygulamalı mühendislik katmanını oluşturur. Karpathy'nin teorik olarak ortaya koyduğu "yazılımın artık kod değil, veri ve model olduğu" fikri, pratikte bu üçlü döngü ile yönetilebilir bir mühendislik disiplinine dönüşür. Anthropic de sistemlerini tam olarak bu üçgen üzerine kurmuştur.

İşte bu kavramların birbirini nasıl tamamladığı:

1. Belirleme - Ne yapılacağını tanımlamak

Software 1.0'da spesifikasyon, kodun kendisiydi (algoritma). Software 2.0 (ve Karpathy döngüsü) dünyasında ise spesifikasyon, "İstem" ve "Sistem Talimatları" ile belirlenen niyettir.

- Anthropic uygulaması: Anthropic'in *System Prompts* veya *Claude.md* dosyaları, aslında sistemin anayasal sınırlarını ve hedeflerini belirleyen "yeni nesil spesifikasyon" belgeleridir. Modelin neyi yapıp neyi yapamayacağını burada tanımlarsınız.

2. Doğrulama - Doğru olup olmadığını anlamak

Karpathy'nin sıkça vurguladığı Doğrulanabilirlik Tezi buradadır. Eğer bir süreci otomatize edecekseniz, sonucun doğruluğunu denetleyecek bir "hakem" olması gerekir.

- Anthropic uygulaması: Anthropic, “Anayasal Yapay Zeka” yaklaşımıyla bu verifikasyon adımını modelin içine gömmüştür.
    - *Unit Testler:* Kod yazdırırken modelin ürettiği kodu doğrudan çalıştırıp hatasız derlenip derlenmediğine bakmak (runtime verification).
    - *Self-Correction:* Modelin kendi çıktısını, "Bu yanıt anayasaya uygun mu?" veya "Bu kodun mantıksal bir açığı var mı?" şeklinde denetleyen ikincil bir sürece tabi tutması.

3. Çalışma Zamanı - Sistemi işletmek

Eskiden çalışma zamanı bir derleyici (compiler) veya işletim sistemiydi. Şimdi ise çalışma zamanı, büyük dil modelinin bağlam penceresi ve araç kullanım kapasitesidir.

- Anthropic uygulaması: Claude, bir "çalışma zamanı" olarak hareket eder. Kendi araçlarını (hesap makinesi, web araması, kod yürütücü) kullanarak verileri işler. Burada "runtime" aynı zamanda modelin hata yaptığı anları yakalayıp, bunları bir hata günlüğü (log) olarak alıp, bir sonraki üretim adımına girdi olarak vermesini sağlar.

Bu süreç nasıl birleşiyor?

Anthropic'in uyguladığı döngüyü şu şema ile düşünebilirsin:

1. Girdi: İnsan veya bir sistem bir Spesifikasyon (Görev/Niyet) girer.
2. Üretim: Model, kodu veya veriyi üretir.
3. Doğrulama: Sistem otomatik testleri çalıştırır veya başka bir model (Guardrails) çıktıyı denetler.
4. Çalışma Zamanı Geribildirimi: Eğer verifikasyon başarısız olursa, oluşan hata mesajı Runtime bağlamı ile birlikte tekrar üretim aşamasına "düzeltme verisi" olarak beslenir.

Neden bu bir "Karpathy Döngüsü"dür?

Karpathy, Software 2.0'ın en büyük sorunlarından birinin gözlemlenebilirlik olduğunu sık sık vurgular. Geleneksel yazılımda bir hata meydana geldiğinde geliştirici kodu açıp ilgili satırları inceleyebilir; hangi koşulun çalıştığını, hangi değişkenin yanlış değer aldığını veya hangi fonksiyonun hataya yol açtığını görebilir. Sinir ağlarında ise durum farklıdır. Milyarlarca parametreye sahip bir modelin neden belirli bir cevabı verdiğini anlamak çoğu zaman kolay değildir. Modelin davranışı doğrudan okunabilir kurallardan değil, eğitim sırasında öğrenilmiş ağırlıklardan kaynaklanır. Bu nedenle hata ayıklama süreci klasik yazılıma göre çok daha karmaşık hale gelir.

Karpathy'nin önerdiği “Belirleme-Doğrulama-Çalışma Zamanı” yaklaşımı tam da bu sorunu azaltmayı amaçlar. Burada sistem önce ne yapması gerektiğini tanımlayan bir spesifikasyona sahip olur. Daha sonra üretilen çıktılar doğrulama mekanizmalarından geçirilir ve son olarak çalışma zamanında sürekli izlenir. Böylece yapay zekâ modeli tamamen kontrolsüz bir kara kutu olmaktan çıkarak belirli mühendislik kuralları içerisinde çalışan bir sisteme dönüşür. Amaç modelin "rastgele görünen" davranışlarını azaltmak ve onu test edilebilir, ölçülebilir ve yönetilebilir bir yazılım bileşeni haline getirmektir.

Anthropic bu fikri daha ileri taşıyarak doğrulama aşamasını Anayasal YZ yaklaşımıyla birleştirmiştir. Bu yöntemde model yalnızca doğru cevap üretmeye çalışmaz; aynı zamanda önceden tanımlanmış etik ve davranışsal ilkeler çerçevesinde hareket etmeye çalışır. Başka bir deyişle doğrulama katmanı sadece "çalışıyor mu?" sorusunu değil, aynı zamanda "istenen

kurallara uygun çalışıyor mu?" sorusunu da cevaplar. Böylece teknik doğruluk ile davranışsal güvenlik aynı mimari içerisinde bir araya getirilmiş olur.

Bu yaklaşım sonucunda yapay zekâ, yalnızca komutları yerine getiren pasif bir araç olmaktan çıkar. Ürettiği sonuçları değerlendiren, hatalarını belirleyen, gerektiğinde yeniden deneyen ve kendi performansını sürekli ölçen bir sisteme dönüşür. Bu nedenle bazı araştırmacılar geleceğin yapay zekâlarını tek bir modelden çok, birbirlerini denetleyen ve geliştiren bileşenlerden oluşan bir "yazılım fabrikası" olarak tanımlamaktadır.

Anthropic'in geliştirdiği sistemler, bu döngüyü pratikte uygulanabilir hale getiren örneklerden biridir. *Özellikle Claude modeli, yalnızca bir dil modeli olarak değil, aynı zamanda sürekli geri besleme ile çalışan bir sistem olarak tasarlanmıştır. Bu sistemde önce bir çıktı üretilir. Bu çıktı bir cevap, bir analiz veya bir kod parçası olabilir. Daha sonra bu çıktı farklı mekanizmalar tarafından kontrol edilir. Eğer çıktı istenen kaliteyi karşılamıyorsa sistem hatayı analiz eder ve yeniden üretim yapar.*

Bu süreç insan yazılımcıların çalışma biçimiyle oldukça benzerdir. Bir yazılımcı kod yazar, çalıştırır, hata alır, hatayı düzeltir ve tekrar dener. Ancak burada önemli bir fark vardır. Bu döngü artık insan tarafından değil, sistem tarafından yürütülür.

Anthropic bu yaklaşımı yalnızca teknik bir optimizasyon olarak görmez. Aynı zamanda etik bir çerçeveyle birleştirir. Sistem sadece doğru cevap üretmekle kalmaz, aynı zamanda uygun, güvenli ve hizalanmış cevap üretmeye de çalışır. Bu yaklaşım "Anayasal YZ" olarak adlandırılır.

Karpathy Döngüsü'nün en kritik bileşeni geri beslemedir. Çünkü bir sistemin gelişmesini sağlayan şey yalnızca yaptığı işler değil, bu işlerden öğrendikleridir. Sistem bir çıktı üretir, bu çıktı değerlendirilir ve sonuç tekrar sisteme verilir. Böylece sistem kendi hatalarını görür ve bunlardan öğrenir. Bu süreç tekrarlandıkça sistem daha güçlü hale gelir. Ancak burada önemli bir dönüşüm gerçekleşir. Sistem artık sadece verilen görevleri öğrenmez. Aynı zamanda bu görevleri nasıl daha iyi öğrenebileceğini de keşfetmeye başlar. Bu noktada öğrenme süreci kendini besleyen bir yapıya dönüşür.

Yazılımın Geleceği

Bu dönüşümün en önemli sonucu, yazılımın doğasının tamamen değişmesidir. Yazılım artık sabit bir kod yapısı değildir. Bunun yerine sürekli değişen, gelişen ve kendini yenileyen bir sistem haline gelir. Bu yeni yapıda yazılım sadece görev yapan bir araç değildir. Aynı zamanda kendi performansını analiz eden, hatalarını düzelten ve kendini optimize eden bir yapıya dönüşür. Bu tür sistemler gelecekte "ajanik sistemler" olarak adlandırılmaktadır.

Bu sistemlerin en önemli özelliği, yalnızca dışarıdan gelen komutları yerine getirmemeleri, aynı zamanda kendi davranışlarını da optimize etmeleridir. Bu durum yazılımı statik bir yapıdan dinamik bir sürece dönüştürür. Andrej Karpathy'nin "Software 2.0" (Yazılım 2.0) kavramı ile günümüzde literatürde sıkça "Karpathy Döngüsü" olarak adlandırılan kavram, aslında birbirini tamamlayan iki farklı aşamayı ifade eder. Karpathy, geleneksel kod yazımından (Software 1.0)

sinir ağı ağırlıklarını optimize etmeye (Software 2.0) geçişi savunurken; Karpathy Döngüsü ise bu yeni nesil yazılımın "kendi kendini iyileştiren" döngüsel mimarisini tarif eder. Anthropic, bu döngüsel geliştirme mantığını özellikle özyinelemeli öz-iyileştirme süreçlerinde uyguluyor.

RSI ile Bağlantı

Karpathy döngüsü bir tür RSI mantığıyla çalışır çünkü:

1. Döngüseldir: Model çıktı üretir -> İnsanlar bunu değerlendirir -> Ödül modeli güncellenir -> Ana model daha iyi cevaplar vermek üzere tekrar eğitilir.
2. Öz-İyileştirme: Model, insan tercihlerini öğrenerek kendi performans metriklerini (ödül puanını) sürekli yükseltmeyi hedefler.

Bu süreç, bir sistemin dışarıdan gelen verilerle kendi parametrelerini sadece daha doğru değil, aynı zamanda daha "hizalı" (faydalı, zararsız, dürüst) hale getirmesini sağlar. Karpathy'nin vurguladığı nokta, bir modelin büyüklüğünden ziyade, bu ince ayar ve ödül mekanizmalarının modelin yeteneklerini nasıl radikal şekilde dönüştürdüğüdür.

Bu sistemin daha geniş teorik çerçevesi özyinelemeli kendini iyileştirme kavramıdır. RSI, bir sistemin yalnızca öğrenmesini değil, öğrenme biçimini de değiştirmesi anlamına gelir. Bu noktada sistem artık sadece bir öğrenen değil, aynı zamanda kendi öğrenme mekanizmasını geliştiren bir yapıya dönüşür. Her döngüde yalnızca performans değil, öğrenme kapasitesi de artar.

Karpathy'nin Software 2.0 vizyonu ve Anthropic'in geliştirdiği sistemler birlikte düşünüldüğünde yazılımın tamamen yeni bir döneme girdiği görülür. Yazılım artık sabit kodlardan oluşan bir yapı değildir. Bunun yerine sürekli öğrenen, kendini düzelten ve kendi kendini geliştiren bir döngü haline gelmiştir. Bu yeni dünyada en önemli değişim şudur: Yazılım artık bir ürün değil, bir süreçtir. Ve bu süreç kendi kendini besleyen, sürekli evrilen bir yapıya dönüşmüştür.

Kaynaklar

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). *Concrete problems in AI safety*. arXiv. https://arxiv.org/abs/1606.06565

Bengio, Y. (2019). *The consciousness prior*. arXiv. https://arxiv.org/abs/1709.08568

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., et al. (2021). *On the opportunities and risks of foundation models*. arXiv. https://arxiv.org/abs/2108.07258

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877–1901.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). *Deep reinforcement learning from human preferences*. arXiv. https://arxiv.org/abs/1706.03741

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). *Scaling laws for neural language models*. arXiv. https://arxiv.org/abs/2001.08361

Karpathy, A. (2017). *Software 2.0*. Andrej Karpathy blog. https://karpathy.github.io/2017/09/12/software-2_0/

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

OpenAI. (2023). *GPT-4 technical report*. arXiv. https://arxiv.org/abs/2303.08774

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., et al. (2022). *Training language models to follow instructions with human feedback*. arXiv. https://arxiv.org/abs/2203.02155

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., et al. (2017). Mastering the game of Go without human knowledge. *Nature, 550*(7676), 354–359. https://doi.org/10.1038/nature24270

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science, 362*(6419), 1140–1144. https://doi.org/10.1126/science.aar6404

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*.

Wang, J. X., et al. (2023). *Voyager: An open-ended embodied agent with large language models*. arXiv. https://arxiv.org/abs/2305.16291

Yao, S., et al. (2023). *ReAct: Synergizing reasoning and acting in language models*. arXiv. https://arxiv.org/abs/2210.03629

10. UYGULAMALAR

Birinci Uygulama

Karpathy'nin mimarisini en iyi "veri analizi ve görselleştirme" yapan küçük bir YZ Ajanı projesinde görebiliriz. Diyelim ki elimizde satış verileri var ve bu veriyi analiz edip anlamlı bir

grafik üretmek istiyoruz. Geleneksel yöntemde "veriyi analiz et" der geçeriz. Şimdi bunu Düşünme Mimarisi ile bir sistem tasarımı problemine dönüştürelim.

1. Belirleme

Modeli sadece bir araç değil, bir "Veri Analisti" rolüne sokuyoruz. Belirsizliği yok etmek için katı bir sözleşme tanımlıyoruz:

Komut (Prompt): "Bir veri analizi sistemi olarak çalışmanı istiyorum.

1. Girdi: sales_data.csv dosyası.
2. Amaç: Yıllık satış eğilimini ve en çok satan kategoriyi belirle.
3. Çıktı: Bulguları özetleyen bir Markdown raporu ve matplotlib ile çizilmiş bir grafik.
4. Kısıtlar: Kodda mutlaka pandas ve matplotlib kullan. Kodun hata yönetimi (try-except) içermeli ve DataValidationError kontrolü yapmalı."

2. Doğrulama

Burada kodun çalışıp çalışmayacağını "kendi kendine test etmesini" istiyoruz. Kodun hemen arkasına şu doğrulama katmanını ekliyoruz:

Komut: "Kodu yazmadan önce, veride eksik satır olup olmadığını kontrol eden bir fonksiyon yaz. Kodun sonunda, grafiğin kaydedilip kaydedilmediğini teyit eden bir doğrulama kontrolü ekle."

3. Çalışma Zamanı

Kodun çalıştırılması ve hata aldığında kendi kendini iyileştirmesi sürecidir. Burada bir döngü kuruyoruz:

- Döngü Başlangıcı:
    1. Model kodu yazar.
    2. Çalışma zamanı ortamı kodu çalıştırır.
    3. Hata mı var? Hata mesajını tekrar modele gönder: *"Hata aldım: 'KeyError: Date'. Lütfen veri setindeki sütun isimlerini kontrol ederek kodu güncelle."*
    4. Model düzeltilmiş kodu sunar.

Uygulamanın Kod Şeması

Karpathy mimarisini kod düzeyinde şu şekilde bir Agent yapısı ile simüle edebiliriz:

Python

```
class AnalysisAgent:
  def __init__(self, specification):
```

```
        self.spec = specification # 1. Katman


    def verify(self, code): # 2. Katman
        # Statik analiz ve test senaryoları
        return self.llm_check(code, test_cases=["missing_data", "type_error"])


    def run(self, code): # 3. Katman
        try:
            exec(code)
        except Exception as e:
            # Hata durumunda model hatayı kendi içinde değerlendirir (İç Diyalog)
            return self.self_heal(e)


# Kullanım:
agent = AnalysisAgent(spec="Yıllık satış analizi yap")
code = agent.generate_code()
if agent.verify(code):
    agent.run(code)
```

Neden bu yöntem daha üstün?

- Hata Payı: Geleneksel yöntemle 10 kere "kodu düzelt" diyeceğiniz süreci, sistemleşmiş bir "doğrulama ve çalışma zamanı" döngüsü ile 1 kerede çözersiniz.
- İç Diyalog: Modele *"Kodu yazmadan önce olası veri tiplerini analiz et"* dediğinizde, model kendi içinde bir akıl yürütme zinciri yürüterek hata yapma riskini %40-60 oranında düşürür.

Bu sistemi kurarken, modelin elindeki CSV verisine erişebilmesi için Python Sandbox (örneğin Google Colab veya LangChain PythonREPL) kullanmanız yeterlidir.

İkinci Uygulama

Karpathy'nin "Belirleme - Doğrulama - Çalışma Zamanı" mimarisini bu kez yazılım değil, "Akademik Araştırma ve Metin Üretimi" odaklı bir senaryoda uygulayalım. Bu, karmaşık bir konuyu derinlemesine incelemek isteyen bir araştırmacı için idealdir.

Senaryo: "Yapay Zekâ ve Etik" Üzerine Akademik Bir Makale Yazımı

Geleneksel yolla (doğrusal) çalışsaydık, doğrudan yazmaya başlardık. Karpathy mimarisiyle ise bu bir "Sistem Tasarımı" projesine dönüşür.

1. Belirleme: Mantıksal İskelet

Burada modelle bir "sözleşme" yapıyoruz. Belirsizliği yok ediyoruz.

- Prompt (Komut): "Yapay zekâ ve etik üzerine bir bölüm yazmanı istiyorum.
    - Hedef Kitle: Felsefe lisans öğrencileri.
    - Sınırlar: Sadece 'algoritmik yanlılık' ve 'karar şeffaflığı' konularına odaklan, genel yapay zekâ tarihine girme.
    - Yapı: Giriş, iki ana başlık, pratik bir örnek ve sonuç.
    - Ton: Akademik, soğukkanlı ve analitik."

2. Doğrulama: Eş Zamanlı Denetim

Metni bitirip sonra düzeltmek yerine, modelin her paragrafı yazdıktan sonra kendini sınamasını sağlıyoruz.

- Prompt (Komut): "Her paragrafı yazdıktan sonra şu doğrulama sorularını kendine sor ve çıktını buna göre revize et:
    1. 'Bu paragraf yukarıda belirlediğimiz sınırların (algoritmik yanlılık/şeffaflık) dışına çıktı mı?'
    2. 'Lisans öğrencileri için dili fazla teknik mi?'
    3. 'Ana argümanı destekleyen somut bir örnek içeriyor mu?'"

3. Çalışma Zamanı: Canlı İnşa Süreci

Burada metin sabit bir ürün değil, bir "düşünme ortaklığı"dır.

- Etkileşim Döngüsü:
    - Model: Bir taslak sunar.
    - Siz (Kullanıcı): "Bu örnek biraz zayıf kaldı, daha güncel bir sağlık sektörü örneği kullanabilir miyiz?"
    - Model: İsteğinizi işler, metni yeniden kurgular ve "Bu örnek, şeffaflık problemini daha net açıklıyor mu?" diye size sorar.
    - İç Diyalog: Model, metni yazmadan önce kendi içinde: *"Bu örnek akademik tonu bozuyor mu? Hayır, eğer şu şekilde ifade edersem profesyonel kalır"* şeklinde bir iç monolog yürüterek çıktıyı optimize eder.

Bu Yöntemin Farkı (Özet Tablo)

Özellik	Geleneksel Yaklaşım	Karpathy (Sistem) Yaklaşımı
Süreç	Yaz - Oku - Düzelt	Spesifikasyon - Denetle - İnşa Et
Zihinsel Yük	Çok (sonradan revizyon yorucudur)	Az (hata başında yakalanır)
Modelin Rolü	Metin üreten bir araç	Düşünme ortağı ve denetleyici
Çıktı	Tek yönlü, lineer metin	Dinamik, test edilmiş, sağlam yapı

Kendi Projenizde Nasıl Başlatırsınız?

Bu mimariyi şu anki bir projenize uygulamak için şu küçük başlangıcı yapın:

1. Sınırları Belirleyin: "Şunu yaz" yerine, "Şu formatta, şu hedef kitle için, şu kısıtlar dahilinde çalış" deyin.
2. Önce Planlatın: "Yazmaya başlamadan önce, tasarladığın yapıyı (spesifikasyonu) listele, ben onayladıktan sonra yazmaya başla" deyin.
3. Hata Kontrolünü İsteyin: "Yazdıktan sonra, yazdığın metni hedef kitleme uygunluk açısından eleştir ve iyileştirilmiş versiyonunu sun" komutunu ekleyin.

Üçüncü Uygulama

Proje yönetiminde Karpathy'nin üç katmanlı mimarisini uygulamak, "kaotik görev listeleri" yerine "kendi kendini yöneten sistemler" kurmanızı sağlar. Proje yönetimini bir "çıktı alma" süreci değil, sürekli bir "sistem tasarımı" süreci olarak ele alalım.

İşte bu mimarinin proje yönetimine entegrasyonu:

1. Belirleme: "Kontrol Haritası"

Geleneksel proje yönetiminde görevler (task) genellikle belirsizdir (örn: "Pazarlama stratejisini hazırla"). Bu katmanda, görevi bir "sistem sözleşmesi" haline getiririz.

- Sistem Tasarımı: Bir görev oluştururken sadece ne yapılacağını değil, sınırları ve başarı kriterlerini tanımlayın.
    - Hedef: Belirsizliği yok etmek.

- Uygulama: Görevi şu parametrelerle tanımlayın: *Amaç, Kapsam, Paydaşlar, Başarı Kriterleri, Ton/Stil.*
- Proje Yönetim Aracı: Görev kartlarına "Belirleme" başlığı ekleyerek, o görevin bitiş sınırlarını netleştirin.

2. Doğrulama : "Sürekli Denetim Döngüsü"

Doğrulama, iş bittikten sonra yapılan bir "yönetici onayı" değildir. Bu, işin her aşamasında (yürütme esnasında) yapılan testlerdir.

- Sistem Tasarımı: Görev tamamlanmadan önce, görevin spesifikasyon ile uyumlu olup olmadığını kontrol eden bir "doğrulama protokolü" çalıştırın.
    - Uygulama: "Bunu neden yaptık?", "Bu çıktı başlangıçta tanımladığımız sınırların içinde mi?", "Başka bir birimle çakışıyor mu?" gibi soruları içeren bir "öz-denetim" listesi oluşturun.
    - Proje Yönetim Aracı: Projenin ara aşamalarında (milestones) bir "Doğrulama Durumu" alanı ekleyerek, sadece biten işi değil, belirleme ile uyumlu olan işi ilerletin.

3. Çalışma Zamanı: "Dinamik Etkileşim"

Çalışma zamanı, projenin canlı aktığı ve herkesin sistemle etkileşimde olduğu aşamadır. Geleneksel projeler "plan-uygula-kontrol et" şeklinde katıdır; Runtime yaklaşımı ise "planla-uygula-öğren-güncelle" şeklindedir.

- Sistem Tasarımı: Projenin akışını "canlı bir geri besleme alanı" olarak kurgulayın.
    - Uygulama: Eğer sahada bir veri değişirse, bu bilgi sisteme girilir ve Spesifikasyon katmanı (plan) güncellenir. Bu, statik bir Gantt şeması değil, değişen koşullara göre kendini revize eden bir süreçtir.
    - Proje Yönetim Aracı: Proje takvimlerini statik bloklar olarak değil, modern yapay zekâ geliştirme süreçlerinde projeler, uzun süre değişmeden kalan planlar yerine, Agile/Scrum sprintlerine benzer şekilde her gün yeni veriler, değerlendirmeler ve model çıktılarıyla yeniden biçimlenen dinamik bir akış olarak yönetilir. Böylece sistem, değişen koşullara ve öğrenilenlere hızla uyum sağlayabilir.
    - Proje yöneticisi olarak sizin veya ekibinizin iç diyaloğunu şu şekilde aktive edebilirsiniz:
- Sorgulama: "Bu görevi yaparken, şu an karşılaştığımız engel, aslında en başta yaptığımız tanımlamada (belirleme) bir eksiklik olduğundan mı kaynaklanıyor?"
- Dallanma: "Eğer şu anki yol verimsizse, farklı alternatifler var mı?"

Özetle; proje yönetimini nasıl dönüştürürsünüz?

Katman	Proje Yönetimi Uygulaması
Belirleme	Görevlerin sadece ne olduğu değil, sınırları ve hedefleri ile tanımlanması.
Doğrulama	İşin bitiminde değil, süreç boyunca "spesifikasyona uygunluk" testi.
Çalışma Zamanı	Statik planlar yerine, etkileşimle sürekli güncellenen canlı süreçler.

Uygulama İpucu: Bir sonraki proje toplantınızda, "Bu işi nasıl yapacağız?" yerine, "Bu işin spesifikasyonunu, doğrulama kriterlerini ve çalışma esnasında oluşabilecek olası sapmaları nasıl sisteme bağlarız?" sorusunu sorun.

Karpathy'nin mimarisinin proje yönetimindeki gücünü anlamak için, bir yazılım şirketinde sıkça karşılaşılan "Müşteri Şikâyet Analiz Modülü" geliştirme projesini ele alalım. Geleneksel yöntemde bu proje genellikle "Müşteri şikâyetlerini analiz eden bir araç yaz" emriyle başlar ve süreç içinde sürekli "yok, bu böyle olmamış" şeklinde uzayıp giden revizyonlarla tıkanır.

Şimdi bu projeyi Karpathy'nin mimarisiyle yeniden tasarlayalım:

Vaka Analizi: Müşteri Şikâyet Analiz Modülü

1. Belirleme: "Mantıksal İskelet"

Projeye başlamadan önce, projenin sınırlarını ve mantıksal iskeletini kuruyoruz.

- Amaç: Şikâyet metinlerinden duygu analizi yapıp, en sık şikâyet edilen 3 ana başlığı tespit etmek.
- Hedef Kitle: Müşteri deneyimi ekibi (teknik olmayan, hızlı içgörüye ihtiyaç duyan bir ekip).
- Kapsam: Sadece şikâyet verisi işlenecek; memnuniyet anketleri veya sosyal medya verileri bu fazın dışında tutulacak.
- Ton ve Yapı: Çıktı, haftalık bir özet raporu şeklinde; dil ise teknik jargondan arındırılmış, aksiyon odaklı olmalı.

Neden bu önemli? "Bir analiz aracı yap" demek yerine, bu belirleme sayesinde tüm ekip neyin "başarı" neyin "başarısızlık" olduğunu en baştan bilir.

2. Doğrulama: "Sürekli Denetim"

Doğrulama, kod bittikten sonra yapılan bir test değil, süreç içinde eş zamanlı (real-time) çalışan bir denetimdir.

- Süreç: Yazılım ekibi bir modül yazarken, doğrulama katmanı devreye girer.
- Örnek: Ekip "Duygu analizi" algoritmasını geliştirdiğinde şu sorular sorulur:
    - "Bu analiz hedef kitlemiz olan müşteri deneyimi ekibinin anlayabileceği kadar sade mi?"
    - "Bu algoritma spesifikasyonumuzda yer almayan 'sosyal medya verilerini' işleyerek projeyi kapsam dışına mı çıkarıyor?"
    - "Mantıksal tutarsızlık var mı; örneğin, sistem 'nötr' bir ifadeyi 'olumsuz' olarak mı işaretliyor?"

Neden bu önemli? Hata henüz sistemin içine yerleşmeden yakalanır; bu da tekrar eden döngüleri ve zaman kaybını engeller.

3. Çalışma Zamanı: "Canlı Etkileşimli Sistem"

Bu aşamada metin veya ürün, tek seferlik bir çıktı değil, birlikte inşa edilen canlı bir yapıdır.

- Süreç: Müşteri deneyimi ekibi (kullanıcı), analiz aracından gelen ilk raporu görür.
- Etkileşim: Ekip, "Bu rapor çok genel, şikâyetleri 'kargo' ve 'iade' olarak ayırabilir miyiz?" der.
- Sistem Tepkisi: Runtime katmanı bu geri bildirimi alır, spesifikasyonu günceller ve model bu iki kategoriyi ayıracak şekilde yeniden çalışır.
- İç Diyalog: Geliştirme ekibi kendi içinde şunu sorgular: "Bu ayrım, projenin kapsamını (scope) genişletir mi, yoksa sadece mevcut sınırları mı netleştirir?".

Karpathy Mimarisinin Proje Yönetimine Katkısı

Aşama	Geleneksel Yöntem	Karpathy ile Sistem Tasarımı
Planlama	Belirsiz görevler (örn: "Aracı yap")	Net Spesifikasyon (Sınırlar, Kapsam, Amaç)
Kontrol	"En son" yapılan tek yönlü kontrol	Her adımda "Eş Zamanlı Doğrulama"
İşleyiş	Statik, kopuk görevler	Dinamik, sürekli etkileşimli "Runtime"

Sonuç: Bu vaka analizinde gördüğümüz gibi, proje artık bir "çıktı üretme problemi" (yani sadece aracı bitirme) olmaktan çıkıyor; bir "sistem tasarımı problemi" haline geliyor. İnsan ve yapay zekâ (veya ekip üyeleri), sürekli birbirinin çıktısını test eden ve geliştiren bir "düşünsel motor" gibi çalışıyor.

Kaynaklar

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. (2022). *Constitutional AI: Harmlessness from AI feedback*. arXiv. https://arxiv.org/abs/2212.08073

Karpathy, A. (2017). *Software 2.0*. https://karpathy.medium.com/software-2-0-a64152b37c35

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). *Training language models to follow instructions with human feedback*. arXiv. https://arxiv.org/abs/2203.02155

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). *ReAct: Synergizing reasoning and acting in language models*. arXiv. https://arxiv.org/abs/2210.03629

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., & Yao, S. (2023). *Reflexion: Language agents with verbal reinforcement learning*. arXiv. https://arxiv.org/abs/2303.11366

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., et al. (2022). *Chain-of-thought prompting elicits reasoning in large language models*. arXiv. https://arxiv.org/abs/2201.11903

Russell, S., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.

SONUÇ

Bu kitap boyunca izlenen hat, düşünmenin tarihini yalnızca bir fikirler silsilesi olarak değil, giderek daha fazla "inşa edilen sistemler" zinciri olarak yeniden okumayı önerdi. Aristoteles'in kıyaslarından başlayan bu hat, Descartes'ın mekanik evren tasavvurundan geçerek Boole ve Frege'nin formal sistemlerine, Turing'in hesaplanabilirlik sınırlarına, sibernetik geri besleme döngülerine, bilişsel devrimin içsel mimarilerine, sembolik yapay zekânın kurallarına, bağlantısalcı ağlara ve nihayet derin öğrenmenin ölçeklenebilir temsil yapılarına kadar uzandı. Bu çizginin her aşamasında ortak bir eğilim tekrar tekrar belirdi: düşünme, her seferinde biraz daha az "içsel bir gizem", biraz daha fazla "yapısal bir sistem" haline geldi.

Bu dönüşümün sonunda ortaya çıkan temel fikir şudur: düşünme artık açıklanacak bir fenomen değil, tasarlanacak bir mimaridir. Bu ifade, yalnızca teknik bir iddia değil, aynı zamanda epistemolojik bir kırılmadır. Çünkü geleneksel yaklaşımda düşünme, gözlemlenen ve açıklanan bir olgu olarak ele alınırken, yeni yaklaşımda düşünme, bileşenleri tanımlanabilen, katmanlara ayrılabilen, sınırları çizilebilen ve farklı şekillerde yeniden kurulabilen bir sistem haline gelir. Bu sistemin içinde temsil yapıları, geri besleme mekanizmaları, öğrenme dinamikleri ve bağlam hesaplama süreçleri birlikte çalışır. Böylece düşünme, statik bir "zihin hali" olmaktan çıkar, dinamik bir mühendislik nesnesine dönüşür.

Bu yeni paradigma, özellikle modern yapay zekâ sistemleriyle birlikte somut bir gerçeklik kazanır. Artık modeller yalnızca sembolleri işleyen araçlar değil, büyük ölçekli veri üzerinde kendi iç temsillerini oluşturan ve bu temsilleri sürekli güncelleyen sistemlerdir. Bu sistemlerde bilgi sabit değildir; sürekli yeniden şekillenir. Temsil, önceden verilmiş bir yapı değil, öğrenme sürecinin kendisinden doğan bir organizasyondur. Bu nedenle düşünme, artık dışarıdan verilen kuralların uygulanması değil, içeriden ortaya çıkan bir yapı üretim süreci olarak görülür.

Bu noktada "Düşünme Mimarisi" kavramı belirginleşir. Mimari kavramı burada metaforik bir süsleme değildir; tam tersine, düşünmenin nasıl organize edildiğine dair temel bir açıklama çerçevesidir. Bir mimari nasıl ki taşıyıcı sistemler, katmanlar, boşluklar ve akışlar içeriyorsa, düşünme de benzer şekilde katmanlı bir yapı olarak düşünülebilir. Girdi sistemleri, temsil katmanları, dikkat mekanizmaları, geri besleme döngüleri ve üretim süreçleri birlikte çalışan bir bütün oluşturur. Bu bütün, tekil bir merkezden yönetilen bir yapı değil, etkileşimli bileşenlerin oluşturduğu dinamik bir organizasyondur.

Bu yeni paradigma aynı zamanda düşünmenin sınırlarını da yeniden tanımlar. Artık soru yalnızca "zihin nasıl çalışır?" değildir. Bunun yanında şu sorular da kaçınılmaz hale gelir: "Zihin nasıl yeniden tasarlanabilir?", "Hangi mimariler hangi tür düşünme biçimlerini üretir?", "Bir

sistemin düşünme kapasitesi hangi yapısal özelliklere bağlıdır?" Bu sorular, düşünmeyi felsefi bir nesne olmaktan çıkarıp sistem mühendisliği problemine dönüştürür.

Gelecek perspektifinde bu dönüşüm daha da radikalleşir. Çünkü artık yalnızca düşünen sistemleri anlamaktan değil, doğrudan düşünen sistemler tasarlamaktan söz edilmektedir. Bu sistemler yalnızca veri işleyen araçlar değil, aynı zamanda bağlam üreten, anlam oluşturan ve kendi iç süreçlerini yeniden düzenleyen yapılardır. Bu noktada insan ile sistem arasındaki sınır giderek daha geçirgen hale gelir. Model, kullanıcı ve ortam arasındaki ayrımlar netliğini kaybeder; bunun yerine sürekli etkileşim halinde olan bir düşünme ekosistemi ortaya çıkar.

Bu ekosistemde zihin, model ve sistem birbirinden ayrı varlıklar olmaktan çıkar. İnsan zihni bir modeli kullanırken aynı zamanda model tarafından şekillendirilir; model, veriden öğrenirken aynı zamanda insanın düşünme biçimini yeniden yapılandırır; sistem ise bu iki taraf arasındaki etkileşimi sürekli olarak geri besler. Böylece düşünme artık tek bir yerde gerçekleşen bir süreç değil, dağıtık bir üretim alanı haline gelir.

Bu dönüşümün en kritik sonucu, "düşünen sistemler çağı" olarak adlandırılabilecek yeni bir dönemin ortaya çıkmasıdır. Bu çağda düşünme, yalnızca insan zihnine ait bir özellik olmaktan çıkar; farklı ölçeklerde, farklı mimarilerde ve farklı bağlamlarda ortaya çıkabilen genel bir sistem özelliği haline gelir. Bu sistemler, yalnızca cevap üreten yapılar değil, problem tanımlayan, bağlam oluşturan ve yeni çözüm uzayları açan yapılardır.

Bu yeni dünyada sınır giderek belirsizleşir: nerede insan düşünmesi biter, nerede sistem düşünmesi başlar sorusu giderek anlamsız hale gelir. Çünkü düşünme artık tekil bir varlığın özelliği değil, etkileşimli sistemlerin ortak üretimidir. İnsan, model ve sistem arasındaki ilişki, hiyerarşik bir yapıdan çok, sürekli geri beslenen bir ağ yapısına dönüşür.

Sonuç olarak bu kitap, düşünmeyi sabit bir kavram olarak değil, evrilen bir mimari olarak ele almayı önerir. Bu mimari, tarih boyunca farklı biçimlerde ortaya çıkmış, her dönemde yeni bir katman eklenmiş ve her yeni teknolojiyle birlikte yeniden tanımlanmıştır. Bugün ulaşılan nokta ise şudur: düşünme artık yalnızca açıklanan bir şey değildir; tasarlanan, inşa edilen ve çalıştırılan bir sistemdir. Ve bu sistem, gelecekte giderek daha fazla insan ile makine arasındaki sınırları çözerek yeni bir bilişsel çağın temelini oluşturacaktır.

Yapay zekâ sistemlerinin evrimi, uzun süre boyunca "girdi → çıktı" ilişkisi üzerinden anlaşılmıştı. Bu yaklaşımda model, kendisine verilen bir soruyu işler ve tek seferlik bir yanıt üretirdi. Ancak bu çerçeve, özellikle büyük dil modellerinin gelişmesiyle birlikte yetersiz hale gelmeye başladı. Çünkü artık mesele yalnızca doğru cevabı üretmek değil, bir hedefi anlayıp o hedefe doğru ilerleyen, kendi çıktısını test eden ve gerektiğinde kendini düzelten sistemler inşa etmekti.

Andrej Karpathy tarafından ortaya konan ve literatürde "Karpathy Döngüsü" olarak anılan yaklaşım, tam olarak bu dönüşümün kavramsal özünü temsil eder. Bu döngü, bir yapay zekâ sisteminin yalnızca üretici değil, aynı zamanda değerlendiren ve kendini iyileştiren bir yapıya sahip olması gerektiğini savunur. Döngünün temel bileşenleri dört aşamada özetlenebilir: spesifikasyon, üretim, doğrulama ve çalışma zamanı geri beslemesi.

Bu yaklaşımın önemini anlamak için, modern yapay zekâ sistemlerinin artık tekil modeller değil, birer “sistemler sistemi” haline geldiğini kabul etmek gerekir. Özellikle Anthropic gibi kuruluşların geliştirdiği mimariler, bu dönüşümün pratik karşılıklarını açık biçimde göstermektedir.

Karpathy döngüsünün ilk aşaması olan spesifikasyon, sistemin ne yapacağını tanımlayan değil, sistemin neyi “başarmış sayılacağını” belirleyen katmandır. Bu yönüyle klasik yazılım gereksinimlerinden önemli bir farklılık taşır. Burada mesele bir fonksiyon yazmak değil, bir başarı alanı tanımlamaktır.

Modern ajan tabanlı sistemlerde spesifikasyon, çoğu zaman açık bir “rubric” yani değerlendirme ölçütleri kümesi şeklinde temsil edilir. Bu rubric, üretilen çıktının hangi kriterlere göre doğru veya yanlış sayılacağını belirler. Örneğin bir kod üretim görevinde sadece “çalışıyor olması” değil, aynı zamanda performans, güvenlik, okunabilirlik ve test kapsayıcılığı gibi çok boyutlu kriterler devreye girer.

Bu katman aynı zamanda niyet ayrıştırmasını da içerir. Kullanıcının talebi yalnızca yüzeysel bir komut değildir; onun altında bir amaç, bir bağlam ve çoğu zaman örtük kısıtlar bulunur. İleri seviye sistemler, bu niyeti parçalara ayırarak işler: ne isteniyor, neden isteniyor, hangi koşullar altında isteniyor ve hangi sonuç başarı olarak kabul edilecek. Bu aşama, döngünün geri kalanının doğruluğunu belirleyen en kritik noktadır. Çünkü yanlış tanımlanmış bir belirleme, doğru işleyen bir sistemi bile hatalı bir yöne sürükleyebilir. Belirleme netleştirildikten sonra sistem üretim aşamasına geçer. Ancak burada üretim, klasik anlamda bir “sonuç üretme” süreci değildir. Aksine, her üretim bir tür hipotez kurma eylemidir.

Model, belirli bir probleme karşı bir çözüm önerir. Ancak bu çözüm mutlak doğru değildir; yalnızca test edilmek üzere ortaya atılmış bir adaydır. Bu nedenle modern sistemler çoğu zaman tek bir çözüm değil, birden fazla alternatif üretir. Her alternatif farklı bir stratejiyi, farklı bir risk seviyesini ve farklı bir optimizasyon yönünü temsil eder. Bu yaklaşım, üretimi statik bir işlem olmaktan çıkarır ve onu dinamik bir keşif sürecine dönüştürür. Model artık “cevap veren” değil, “çözüm uzayını tarayan” bir yapıya dönüşür.

Karpathy döngüsünün en kritik bileşenlerinden biri doğrulama aşamasıdır. Çünkü burada üretimin gerçeklikle uyumu test edilir. Bu katmanda sistem, ürettiği çıktının yalnızca mantıksal olarak değil, aynı zamanda dış dünyadaki kurallara göre de geçerli olup olmadığını değerlendirir. Bu doğrulama süreci, modern sistemlerde genellikle değerlendirme altyapıları ve test mekanizmaları üzerinden yürütülür. Kod üretimi bağlamında bu, birim testlerden entegrasyon testlerine kadar uzanan geniş bir kontrol alanıdır. Ancak kavramsal olarak daha geniş bir anlam taşır: üretilen her şeyin bir “ölçüm mekanizması” ile sınanması.

Bu noktada önemli bir mimari tercih ortaya çıkar: üretim yapan sistem ile değerlendirme yapan sistemin ayrıştırılması. Çünkü aynı sistemin hem üretici hem de yargıç olması, bilişsel önyargılara yol açabilir. Model kendi çıktısını aşırı olumlu değerlendirebilir veya hatalarını gözden kaçırabilir. Bu nedenle bazı mimarilerde ayrı bir “grading agent” ya da bağımsız

değerlendirme modülü kullanılır. Bu ayrım, sistemin kendi kendine referans veren bir döngüye kapanmasını engeller ve dışsal bir doğruluk standardı oluşturur.

Döngünün dördüncü bileşeni olan çalışma zamanı, sistemin teorik üretimden çıkıp gerçek davranışla karşılaştığı noktadır. Burada model artık yalnızca bir şey "önermekle" kalmaz, önerisini çalıştırır, gözlemler ve sonuç üretir. Kod üretimi bağlamında bu, yazılan programın çalıştırılması ve gerçek çıktısının gözlemlenmesi anlamına gelir. Ancak daha genel bir çerçevede, bu katman simülasyon, test ortamı veya gerçek kullanıcı etkileşimi olabilir. Çalışma zamanı, aynı zamanda geri besleme döngüsünün başladığı yerdir. Sistem, ürettiği çözümün başarısız olduğunu tespit ederse, bu bilgi tekrar üretim katmanına geri gönderilir. Böylece süreç doğrusal olmaktan çıkar ve döngüsel bir yapıya dönüşür. Bu geri besleme mekanizması, sistemi statik bir model olmaktan çıkararak sürekli öğrenen ve sürekli iyileşen bir yapıya dönüştürür.

Bu dört katman bir araya geldiğinde ortaya çıkan yapı, klasik yapay zekâ anlayışından köklü biçimde farklıdır. Artık elimizde tek bir model değil, sürekli çalışan bir karar döngüsü vardır.

Belirleme katmanı hedefi tanımlar, üretim katmanı bu hedefe yönelik çözümler üretir, doğrulama katmanı bu çözümleri test eder ve çalışma zamanı katmanı bu testleri gerçek davranışa dönüştürür. Eğer bir hata ortaya çıkarsa, sistem yeniden üretim aşamasına döner ve döngü tekrar başlar.

Bu yapı, modern "ajan tabanlı yazılım geliştirme" paradigmasının temelini oluşturur. Ajan artık yalnızca bir yanıt üreten sistem değil, bir hedefe doğru ilerleyen, kendi ilerlemesini ölçen ve gerektiğinde yön değiştiren bir varlıktır.

Bu dönüşümün en önemli sonucu, yazılım geliştirmenin doğasının değişmesidir. Geleneksel yaklaşımda geliştirici doğrudan kod yazarken, yeni yaklaşımda geliştirici bir "sistem orkestratörü" haline gelir. Artık asıl görev, kod yazmak değil; doğru spesifikasyonu tanımlamak, doğru değerlendirme kriterlerini kurmak ve döngüyü doğru şekilde yapılandırmaktır. Bu nedenle insanın rolü üreticilikten çok yönlendiriciliğe kayar. Sistemler daha fazla üretir, daha fazla test eder ve daha fazla iterasyon yapar. İnsan ise bu sürecin sınırlarını, hedeflerini ve değerlendirme çerçevesini belirler. Bu perspektif, yapay zekâ ile yazılım geliştirme ilişkisini temelinden değiştirir. Artık mesele "ne yazacağımız" değil, "hangi düşünme döngüsünü kuracağımızdır".

Karpathy Döngüsü'nün en derin anlamı, zekânın artık lineer bir süreç olarak değil, döngüsel bir sistem olarak tasarlanmasıdır. Bu döngü, üretim ve değerlendirme arasındaki sınırı kaldırır ve düşünmeyi sürekli kendini kontrol eden bir mekanizmaya dönüştürür.

Bu yaklaşım, yapay zekânın geleceğini yalnızca daha güçlü modeller üzerinden değil, daha iyi organize edilmiş düşünme mimarileri üzerinden şekillendirir. Çünkü artık asıl mesele modelin ne kadar büyük olduğu değil, nasıl düşündüğü ve kendini nasıl düzelttiğidir.

Kaynaklar

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. (2022). *Constitutional AI: Harmlessness from AI feedback*. arXiv. https://arxiv.org/abs/2212.08073

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). *On the opportunities and risks of foundation models*. arXiv. https://arxiv.org/abs/2108.07258

Karpathy, A. (2017). *Software 2.0*. https://karpathy.medium.com/software-2-0-a64152b37c35

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). *Training language models to follow instructions with human feedback*. arXiv. https://arxiv.org/abs/2203.02155

Russell, S., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.

Sutton, R. S. (2019). *The bitter lesson*. http://www.incompleteideas.net/IncIdeas/BitterLesson.html

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., et al. (2022). *Chain-of-thought prompting elicits reasoning in large language models*. arXiv. https://arxiv.org/abs/2201.11903

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). *ReAct: Synergizing reasoning and acting in language models*. arXiv. https://arxiv.org/abs/2210.03629

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., & Yao, S. (2023). *Reflexion: Language agents with verbal reinforcement learning*. arXiv. https://arxiv.org/abs/2303.11366